Front-end

JavaScript

JS. Хранение данных

- Хранение данных

JS. Хранение данных

При разработке современных веб-сайтов достаточно часто необходимо сохранять данные на стороне клиента. Для таких целей в браузере существует несколько разных способов, которые появились в разное время и отличаются друг от друга.

JS. Хранение данных

Cookie

Первый способ — это хранение данных в cookie. Этот способ — не просто хранилище данных в браузере. Данные, хранящиеся в куках, также передаются на сервер в виде HTTP-заголовка и могут быть им изменены. Cookie являются частью спецификации протокола HTTP, и их поддерживают все браузеры.

У хранения данных в Cookie есть много ограничений. Они передаются при каждом запросе к серверу, их размер ограничен 4096 байтами, а содержимое должно быть закодировано и быть безопасным, чтобы не сломать отправляемый запрос. Cookie не являются постоянным хранилищем, срок хранения данных по умолчанию ограничен длинной сессии, и для продления хранения кук используется дополнительный параметр.

Cookie имеют не самый удобный интерфейс чтения и записи, поэтому в работе многие разработчики чаще используют сторонние библиотеки, а не взаимодействуют с интерфейсом напрямую.

Как правило, этим способом пользуются для хранения авторизационных данных или когда доступ к записанным данным нужен на сервере. Ещё куки используются для отслеживания поведения пользователя на сайте, но браузеры активно с этим борются.

JS. Хранение данных

Web Storage

Web Storage – это интерфейс взаимодействия с хранилищем. Есть две реализации этого API: Local Storage и Session Storage. Оба способа имеют идентичный API и ограничения, а основным различием является время хранения данных.

Session Storage похож на краткосрочные Cookie, потому что данные в этом хранилище хранятся только во время жизни текущей сессии.

Local Storage, в теории, является бессрочным хранилищем данных. Хотя данные и должны храниться бессрочно, браузеры все равно вводят свои ограничения.

JS. Хранение данных

SessionStorage хранит данные только во время текущей сессии (для вкладки, пока она открыта). Закрытие вкладки или браузера приводит к очищению этих данных. При этом данные сохраняются при обновлении страницы

JS. Хранение данных

Так, например, при переполнении хранилища оно полностью очищается. А Safari очищает Local Storage если к нему не обращались в течение семи дней. То есть если пользователь не посещал ваш сайт больше 7 дней, то данные хранящиеся в нем будут удалены.

Максимальный объем хранимых данных — 5 Мб. При этом любой скрипт, загруженный на странице, может иметь доступ к Web Storage на этой странице. Поэтому не стоит хранить там приватную информацию или токены авторизации, так как таким скриптом может являться вредное браузерное расширение, которое ворует информацию пользователя.

Интерфейс взаимодействия с этим видом хранилища очень простой и достаточно удобный. Данные хранятся в виде ключ-значение, и функции названы интуитивно понятно.

JS. Хранение данных

Данные записываются в виде строк, поэтому если вы хотите сохранить сложную структуру, то сначала нужно сделать из неё строку с помощью **JSON.stringify()** и только после этого записывать.

Запись в Web Storage является синхронной. Это значит, что на время записи браузер не выполняет другие действия. Поэтому избегайте частых записей в этот вид хранилища.

JS. Хранение данных

Indexed DB

Это самый новый способ хранения данных. При этом он обладает достаточно сложным API. В отличие от всех остальных способов, это API асинхронное.

Данный способ представляет собой API базы данных, что зачастую излишне для веб-приложений. Существует много разных библиотек, упрощающих работу с этим видом хранилища.

JS. Хранение данных

localStorage

Это объект, хранящийся в window, который позволяет долговременно сохранять данные в браузере. Работает как хранилище данных в формате ключ-значение — при сохранении данных мы указываем имя поля, в которое должны быть сохранены данные, и затем используем это имя для их получения.

Значения хранятся в виде строк. При попытке сохранения других типов данных, они будут приведены к строке. Например, если записать число, то при чтении нам вернётся число, записанное в строку.

Не имеет ограничений по времени хранения, может быть очищен пользователем вручную или браузером при переполнении автоматически (браузеры на основе движка WebKit, например Safari, очищают localStorage, если к нему не обращались в течение 7 дней).

Максимальный объем данных ограничен размером 5MB.

JS. Хранение данных

localStorage

Запись

Для записи используйте метод **setItem('ключ', 'значение')**. Он принимает два строковых параметра: ключ, по которому будет сохранено значение, и само значение.

```
window.localStorage.setItem('name', 'John')
```

JS. Хранение данных

localStorage

Чтение

За чтение отвечает **getItem('ключ')** с одним параметром, который указывает на ключ для чтения и возвращает полученное значение из хранилища. Если по этому ключу нет значения, то метод вернёт null.

```
window.localStorage.getItem('name') // вернет ' John'

window.localStorage.getItem('user') // вернет ' null'
```

JS. Хранение данных

localStorage

Удаление

Удаляет запись из хранилища **removeItem('ключ')**. Он успешно выполнится даже если указанного ключа не существует в хранилище.

```
window.localStorage.removeItem('name')

window.localStorage.removeItem('user')
```

Очистка хранилища

Метод clear() очищает хранилище полностью.

```
window.localStorage.clear()
```

JS. Хранение данных

localStorage

Количество полей в хранилище

Используя свойство **length**, можно узнать, сколько всего полей было записано в хранилище.

```
console.log(window.localStorage.length);

// 0
```

Получение ключа по индексу

Метод **key()** получает ключ по индексу. Значения в хранилище хранятся в порядке их добавления, поэтому значение, добавленное первым, будет храниться в позиции 0 и так далее.

```
window.localStorage.setItem('name', 'Дока Дог')

console.log(window.localStorage.key(0));
```

JS. Хранение данных

Объекты localStorage и sessionStorage являются свойствами глобального объекта window. А это значит к ним можно обращаться соответственно как window.localStorage и window.sessionStorage, или просто localStorage и sessionStorage.

Для работы с localStorage и sessionStorage нам доступен одинаковый набор свойств и методов:

getItem(key) – получает значение по ключу (key);

setItem(key, value) – добавляет ключ (key) со значением value (если в хранилище уже есть этот ключ, то в этом случае будет просто обновлено его значение);

removeItem(key) – удаляет ключ (key);

clear() – очищает всё хранилище;

key(index) – получает ключ по индексу (в основном используется в операциях перебора);

length – количество ключей в хранилище;

JS. Хранение данных

localStorage

События

При установке значения в хранилище срабатывает глобальное событие storage, с помощью которого можно отслеживать изменения в хранилище.

Событие происходит только на других открытых страницах текущего сайта.

Событие содержит свойства:

key - ключ, который был изменён (при вызове метода clear(), ключ будет null);

oldValue - старое значение, записанное в поле;

newValue - новое значение, записанное в поле;

url - адрес страницы, на которой вызвано изменение.

```
window.addEventListener('storage', function (evt) {
  console.log(evt)
})
```

JS. Хранение данных

Если посмотреть объект event, то среди свойств можно увидеть следующие:

key – ключ, значение которого изменено (возвращает null при очистке хранилища);

oldValue – предыдущее значение (null – если ключ только что был добавлен);

newValue – новое значение (null – при удалении ключа);

storageArea – изменённый объект-хранилище;

url – url документа, в котором произошло обновление хранилища.