

Раздел

[Хранение данных в браузере](#)

Навигация по уроку

Демо localStorage

Доступ как к обычному объекту

Перебор ключей

Только строки

sessionStorage

Событие storage

Итого

Задачи (1)

Комментарии

Поделиться

[Редактировать на GitHub](#) → [Хранение данных в браузере](#) 2-го октября 2020

## LocalStorage, sessionStorage

Объекты веб-хранилища `localStorage` и `sessionStorage` позволяют хранить пары ключ/значение в браузере.

Что в них важно – данные, которые в них записаны, сохраняются после обновления страницы (в случае `sessionStorage`) и даже после перезапуска браузера (при использовании `localStorage`). Скоро мы это увидим.

Но ведь у нас уже есть куки. Зачем тогда эти объекты?

- В отличие от куки, объекты веб-хранилища не отправляются на сервер при каждом запросе. Поэтому мы можем хранить гораздо больше данных. Большинство браузеров могут сохранить как минимум 2 мегабайта данных (или больше), и этот размер можно поменять в настройках.
- Ещё одно отличие от куки – сервер не может манипулировать объектами хранилища через HTTP-заголовки. Всё делается при помощи JavaScript.
- Хранилище привязано к источнику (домен/протокол/порт). Это значит, что разные протоколы или поддомены определяют разные объекты хранилища, и они не могут получить доступ к данным друг друга.

Объекты хранилища `localStorage` и `sessionStorage` предоставляют одинаковые методы и свойства:

- `setItem(key, value)` – сохранить пару ключ/значение.
- `getItem(key)` – получить данные по ключу `key`.
- `removeItem(key)` – удалить данные с ключом `key`.
- `clear()` – удалить всё.
- `key(index)` – получить ключ на заданной позиции.
- `length` – количество элементов в хранилище.

Как видим, интерфейс похож на `Map` (`setItem/getItem/removeItem`), но также запоминается порядок элементов, и можно получить доступ к элементу по индексу – `key(index)`.

Давайте посмотрим, как это работает.

### Демо localStorage

Основные особенности `localStorage`:

- Этот объект один на все вкладки и окна в рамках источника (один и тот же домен/протокол/порт).
- Данные не имеют срока давности, по которому истекают и удаляются. Сохраняются после перезапуска браузера и даже ОС.

Например, если запустить этот код...

```
1 localStorage.setItem('test', 1);
```



...и закрыть/открыть браузер или открыть ту же страницу в другом окне, то можно получить данные следующим образом:

```
1 alert( localStorage.getItem('test') ); // 1
```



Нам достаточно находиться на том же источнике (домен/протокол/порт), при этом URL-путь может быть разным.

Объект `localStorage` доступен всем окнам из одного источника, поэтому, если мы устанавливаем данные в одном окне, изменения становятся

видимыми в другом.

## Доступ как к обычному объекту

Также можно получать/записывать данные, как в обычный объект:

```
1 // установить значение для ключа
2 localStorage.test = 2;
3
4 // получить значение по ключу
5 alert( localStorage.test ); // 2
6
7 // удалить ключ
8 delete localStorage.test;
```

Это возможно по историческим причинам и, как правило, работает, но обычно не рекомендуется, потому что:

1. Если ключ генерируется пользователем, то он может быть каким угодно, включая `length` или `toString` или другой встроенный метод `localStorage`. В этом случае `getItem/setItem` сработают нормально, а вот чтение/запись как свойства объекта не пройдут:

```
1 let key = 'length';
2 localStorage[key] = 5; // Ошибка, невозможно установи
```

2. Когда мы модифицируем данные, то срабатывает событие `storage`. Но это событие не происходит при записи без `setItem`, как свойства объекта. Мы увидим это позже в этой главе.

## Перебор ключей

Методы, которые мы видим, позволяют читать/писать/удалять данные. А как получить все значения или ключи?

К сожалению, объекты веб-хранилища нельзя перебрать в цикле, они не итерируемы.

Но можно пройти по ним, как по обычным массивам:

```
1 for(let i=0; i<localStorage.length; i++) {
2   let key = localStorage.key(i);
3   alert(`${key}: ${localStorage.getItem(key)}`);
4 }
```

Другой способ – использовать цикл, как по обычному объекту `for key in localStorage`.

Здесь перебираются ключи, но вместе с этим выводятся несколько встроенных полей, которые нам не нужны:

```
1 // bad try
2 for(let key in localStorage) {
3   alert(key); // покажет getItem, setItem и другие встр
4 }
```

...Поэтому нам нужно либо отфильтровать поля из прототипа проверкой `hasOwnProperty`:

```
1 for(let key in localStorage) {
2   if (!localStorage.hasOwnProperty(key)) {
3     continue; // пропустит такие ключи, как "setItem",
4   }
5   alert(`${key}: ${localStorage.getItem(key)}`);
6 }
```

Раздел

[Хранение данных в браузере](#)

Навигация по уроку

Демо localStorage

Доступ как к обычному объекту

Перебор ключей

Только строки

sessionStorage

Событие storage

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)

Раздел

[Хранение данных в браузере](#)

Навигация по уроку

Демо localStorage

Доступ как к обычному объекту

Перебор ключей

Только строки

sessionStorage

Событие storage

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)

...Либо просто получить «собственные» ключи с помощью `Object.keys`, а затем при необходимости вывести их при помощи цикла:

```
1 let keys = Object.keys(localStorage);
2 for(let key of keys) {
3   alert(`${key}: ${localStorage.getItem(key)}`);
4 }
```

Последнее работает, потому что `Object.keys` возвращает только ключи, принадлежащие объекту, игнорируя прототип.

## Только строки

Обратите внимание, что ключ и значение должны быть строками.

Если мы используем любой другой тип, например число или объект, то он автоматически преобразуется в строку:

```
1 sessionStorage.user = {name: "John"};
2 alert(sessionStorage.user); // [object Object]
```

Мы можем использовать `JSON` для хранения объектов:

```
1 sessionStorage.user = JSON.stringify({name: "John"});
2
3 // немного позже
4 let user = JSON.parse( sessionStorage.user );
5 alert( user.name ); // John
```

Также возможно привести к строке весь объект хранилища, например для отладки:

```
1 // для JSON.stringify добавлены параметры форматирования
2 alert( JSON.stringify(localStorage, null, 2) );
```

## sessionStorage

Объект `sessionStorage` используется гораздо реже, чем `localStorage`.

Свойства и методы такие же, но есть существенные ограничения:

- `sessionStorage` существует только в рамках текущей вкладки браузера.
- Другая вкладка с той же страницей будет иметь другое хранилище.
- Но оно разделяется между ифреймами на той же вкладке (при условии, что они из одного и того же источника).
- Данные продолжают существовать после перезагрузки страницы, но не после закрытия/открытия вкладки.

Давайте посмотрим на это в действии.

Запустите этот код...

```
1 sessionStorage.setItem('test', 1);
```

...И обновите страницу. Вы всё ещё можете получить данные:

```
1 alert( sessionStorage.getItem('test') ); // после обнов.
```

...Но если вы откроете ту же страницу в другой вкладке и попытаете получить данные снова, то код выше вернёт `null`, что значит «ничего не найдено».

Раздел

[Хранение данных в браузере](#)

Навигация по уроку

Демо localStorage

Доступ как к обычному объекту

Перебор ключей

Только строки

sessionStorage

Событие storage

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Так получилось, потому что sessionStorage привязан не только к источнику, но и к вкладке браузера. Поэтому sessionStorage используется нечасто.

## Событие storage

Когда обновляются данные в localStorage или sessionStorage, генерируется событие [storage](#) со следующими свойствами:

- key – ключ, который обновился (null, если вызван .clear()).
- oldValue – старое значение (null, если ключ добавлен впервые).
- newValue – новое значение (null, если ключ был удалён).
- url – url документа, где произошло обновление.
- storageArea – объект localStorage или sessionStorage, где произошло обновление.

Важно: событие срабатывает на всех остальных объектах window, где доступно хранилище, кроме того окна, которое его вызвало.

Давайте уточним.

Представьте, что у вас есть два окна с одним и тем же сайтом. Хранилище localStorage разделяется между ними.

Вы можете открыть эту страницу в двух окнах браузера, чтобы проверить приведённый ниже код.

Теперь, если оба окна слушают window.onstorage, то каждое из них будет реагировать на обновления, произошедшие в другом окне.

```
1 // срабатывает при обновлениях, сделанных в том же окне
2 window.onstorage = event => {
3   if (event.key !== 'now') return;
4   alert(event.key + ':' + event.newValue + " at " + event.url);
5 };
6
7 localStorage.setItem('now', Date.now());
```

Обратите внимание, что событие также содержит: event.url – url-адрес документа, в котором данные обновились.

Также event.storageArea содержит объект хранилища – событие одно и то же для sessionStorage и localStorage, поэтому event.storageArea ссылается на то хранилище, которое было изменено. Мы можем захотеть что-то записать в ответ на изменения.

**Это позволяет разным окнам одного источника обмениваться сообщениями.**

Современные браузеры также поддерживают [Broadcast channel API](#) – специальный API для связи между окнами одного источника, он более полнофункциональный, но менее поддерживаемый. Существуют библиотеки (полифилы), которые эмулируют это API на основе localStorage и делают его доступным везде.

## Итого

Объекты веб-хранилища localStorage и sessionStorage позволяют хранить пары ключ/значение в браузере.

- key и value должны быть строками.
- Лимит 2 Мб+, зависит от браузера.
- Данные не имеют «времени истечения».
- Данные привязаны к источнику (домен/протокол/порт).

localStorage	sessionStorage
Совместно используется между всеми вкладками и окнами с одинаковым источником	Разделяется в рамках вкладки браузера, среди ифреймов из того же источника
«Переживает» перезапуск браузера	«Переживает» перезагрузку страницы (но не закрытие вкладки)

Раздел

[Хранение данных в браузере](#)

Навигация по уроку

Демо localStorage

Доступ как к обычному объекту

Перебор ключей

Только строки

sessionStorage

Событие storage

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)



API:

- `setItem(key, value)` – сохранить пару ключ/значение.
- `getItem(key)` – получить данные по ключу `key`.
- `removeItem(key)` – удалить значение по ключу `key`.
- `clear()` – удалить всё.
- `key(index)` – получить ключ на заданной позиции.
- `length` – количество элементов в хранилище.
- Используйте `Object.keys` для получения всех ключей.
- Можно обращаться к ключам как к обычным свойствам объекта, в этом случае событие `storage` не срабатывает.

Событие `storage`:

- Срабатывает при вызове `setItem`, `removeItem`, `clear`.
- Содержит все данные об произошедшем обновлении (`key/oldValue/newValue`), `url` документа и объект хранилища `storageArea`.
- Срабатывает на всех объектах `window`, которые имеют доступ к хранилищу, кроме того, где оно было сгенерировано (внутри вкладки для `sessionStorage`, глобально для `localStorage`).

## ✓ Задачи

### Автосохранение поля формы

Создайте поле `textarea`, значение которого будет автоматически сохраняться при каждом его изменении.

Когда пользователь закроет страницу и потом откроет её заново он должен увидеть последнее введённое значение.

Вот пример:

Очистить

[Открыть песочницу для задачи.](#)

решение

Проводим [курсы по JavaScript и фреймворкам](#).



## 💬 Комментарии

перед тем как писать...

Раздел

[Хранение данных в браузере](#)

Навигация по уроку

Демо localStorage

Доступ как к обычному объекту

Перебор ключей

Только строки

sessionStorage

Событие storage

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)

22 Комментариев Learn.JavaScript.RU

Политика конфиденциальности Disqus

Войти ▾



Рекомендовать

Твитнуть

Поделиться

Лучшее в начале ▾



Присоединиться к обсуждению...

войти с помощью

или через DISQUS

Имя

Подписаться

Добавь Disqus на свой сайтДобавить DisqusДобавить

Do Not Sell My Data

© 2007–2020 Илья Кантор | [о проекте](#) | [связаться с нами](#) | [пользовательское соглашение](#) | [политика конфи](#)

