

Трек, альбом, исполнитель 



Главное

Рекомендации

Жанры

Радио

 Моя муз



Главное

ВСЁ

НОВЫЕ РЕЛИЗЫ

ПЛЕЙЛИСТЫ С НОВИНКАМИ

ЧАРТ

НАСТРОЕНИЯ И ЖАНРЫ



2

C is for cookie



HTTP State Management Mechanism

Создание

```
document.cookie = 'subject=math';
```

```
// Перезаписывает значение существующей cookie
```

```
document.cookie = 'subject=physics';
```

```
// Добавляет ещё одну cookie
```

```
document.cookie = 'tips=1';
```

```
document.cookie = 'subject=' + encodeURIComponent('Математика')
```

```
// %D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0
```

УНИКАЛЬНОСТЬ

```
document.cookie = 'subject=math; path=/';
```

```
document.cookie = 'subject=math; domain=ege.yandex.ru';
```

```
document.cookie = 'subject=math; path=/; domain=ege.yandex.ru';
```

key = name + path + domain

domain

```
document.cookie = 'subject=math; domain=ege.yandex.ru';
```

```
GET /
```

```
Host: ege.yandex.ru
```

```
GET /
```

```
Host: old.ege.yandex.ru
```

```
GET /
```

```
Host: yandex.ru
```

```
GET /
```

```
Host: ege.yandex.ru
```

По умолчанию текущий

Можно установить только на текущий или
более специфичный (находясь на yandex.ru
нельзя установить cookie для facebook.com)

path

```
document.cookie = 'subject=math; path=/tests';
```

GET /tests/list

GET /tests/page/2

~~GET /tests list~~

~~GET /~~

expires

```
subject=math; expires=Tue, 19 Apr 2019 00:00:00 GMT
```

Чтобы **удалить**, устанавливаем дату
устаревания в прошлом

subject=math; expires=Tue, 19 Apr 1970 00:00:00 GMT

Чтение

```
document.cookie = 'subject=math; path=/';
```

```
console.log(document.cookie);  
// subject=math
```

Если подходят несколько – доступны все в
порядке от наиболее специфичной к
наименее

Чтение

```
document.cookie = 'subject=math; path=/';
document.cookie = 'subject=physics; path=/oge';

console.log(document.cookie);
// subject=physics; subject=math
```

js-cookie

```
Cookies.set('subject', 'math', { expires: 7, path: ''});
```

```
Cookies.get('subject');
```

```
Cookies.remove('subject');
```

Отправка на сервер

GET / HTTP/1.1

Host: ege.yandex.ru

Cookie: subject=math; tips=1

```
const express = require('express')
const app = express();
const cookieParser = require('cookie-parser');

app.use(cookieParser());

app.use((req, res) => {
    console.log(req.cookies);
    // { subject: 'math', tips: 1 }
});

});
```

Получение с сервера

```
var express = require('express')
var app = express();

app.use((req, res) => {
    res.cookie('subject', 'math', { path: '/tests' });
});
```

HTTP/1.1 200 OK

Set-Cookie: subject=math; path=/tests

HTTP-only

```
res.cookie('subject', 'math', {  
    path: '/tests',  
  
    httpOnly: true  
});
```

HTTP/1.1 200 OK

Set-Cookie: subject=math; path=/tests; **HttpOnly**

Не доступны в js-скриптах на клиенте

Secure

```
res.cookie('subject', 'math', {  
  path: '/tests',  
  
  secure: true  
});
```

HTTP/1.1 200 OK

Set-Cookie: subject=math; path=/tests; **secure**

Не доступны по HTTP

third-party request

```
GET / HTTP/1.1
```

```
Host: awesome-notes.com
```

```
HTTP/1.1 200 OK
```

```

```

```
GET /awesome-cat.jpg HTTP/1.1
```

```
Host: awesome-pics.com
```

third-party request

GET / HTTP/1.1

Host: evil-hacker.com

HTTP/1.1 200 OK

```
<form method="POST" action="https://bank/transferMoney">
    <input name="amount" value="1000">
</form>
<script>document.forms[0].submit();</script>
```

POST /transferMoney HTTP/1.1

Host: bank

amount=1000

third-party cookies

Same-site

```
res.cookie('subject', 'math', {  
  path: '/tests',  
  
  sameSite: 'lax' // strict  
});
```

HTTP/1.1 200 OK

Set-Cookie: subject=math; path=/tests; SameSite=Lax

Cookie не передаются в качестве third-party

Lax

```
<a href="https://yandex.ru/">Открыть Яндекс</a>

<link rel="prerender" href="https://yandex.ru/" />

<form method="GET" action="https://yandex.ru/">

<del form method="POST" action="https://yandex.ru/" />

<iframe src="https://yandex.ru/">

$.get("https://yandex.ru/")

<del img src="https://yandex.ru/logo.png"/>
```

Устаревание

Доступ с сервера

4Kb

Передаются с каждым запросом

Инициализация состояния клиента

Авторизация

Для статики используйте
cookieless домены (CDN)

В cookie храните **id**, по которому можно на сервере получить полные данные

Обfuscate 01100101

Using the Same-Site Cookie Attribute to Prevent CSRF Attacks

HTTP cookie



WebStorage

`LocalStorage` – хранит данные до окончания сессии (закрытие вкладки)

`SessionStorage` – хранит данные
перманентно, пока скрипт или
пользователь не удалит их

Не передаёт данные на сервер

Ограничение в 10Mb

Интерфейс

```
localStorage.setItem('volume', 8);
```

```
localStorage.getItem('volume'); // "8"
```

```
localStorage.removeItem('volume')
```

```
localStorage.clear();
```

```
localStorage.volume = 8;
```

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

```
// QUOTA_EXCEEDED_ERROR
```

Хранит **строки**, а не объекты

```
localStorage.setItem('options', { volume: 8 });
```

```
localStorage.getItem('options'); // "[object Object]"
```

```
localStorage.setItem('options', JSON.stringify({ volume: 8 }));
```

Событие

```
window.addEventListener('storage', event => {
    console.log(event);
});
```

```
{
  key: 'volume',
  oldValue: '8',
  newValue: '6'
}
```

Приватный режим

```
try {
    localStorage.setItem('options', '8');
} catch (error) {
    console.error(error);
    // Error: SecurityError: DOM Exception 18
}
```

Хранение настроек

Хранение промежуточных данных

Строго ограничено источником (origin)

Синхронный интерфейс

Web Storage API

Html5 Local Storage How-To

Client-side Data Storage

WebSQL

Асинхронный интерфейс к SQLite базе

```
const db = openDatabase('my-app', '1.0', null, 1024 * 1024);

db.transaction(tr => {
    tr.executeSql(`create table if not exists notes(
        name TEXT
    )
`);
}

tr.executeSql(`insert into notes(name)
    values("films")
`);
}, console.error);
```

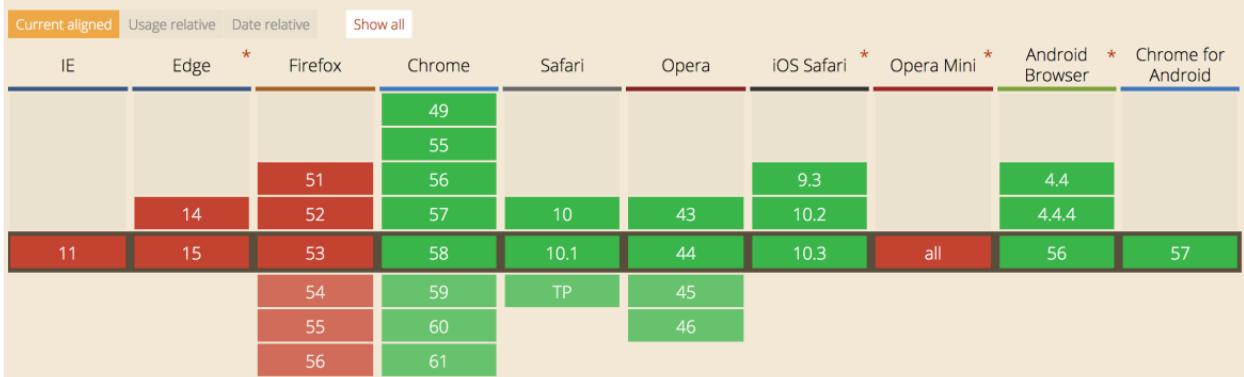
Web SQL Database

■ - UNOFF

Global

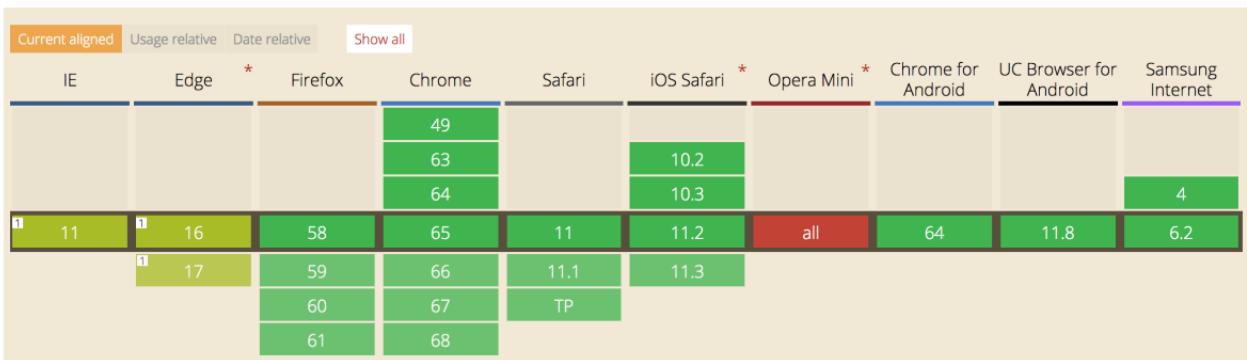
82.31%

Method of storing data client-side, allows Sqlite database queries
for access and manipulation





IndexedDB



Строго ограничено источником (origin)

Нет ограничений на размер^{*}

Асинхронное

Не реляционная, а object storage

He SQL, a API

Создание

```
// Указываем название и версию
const requestDb = indexedDB.open('my-app', 1);

requestDb.onerror = event => {
    console.log(event.target.errorCode);
};

requestDb.onsuccess = event => {
    // Получаем доступ к базе
    const db = event.target.result;
};
```

Обновление

```
// Будет вызван в первый раз и когда сменилась версия
requestDb.onupgradeneeded = event => {
    const db = event.target.result;
    const oldVersion = event.oldVersion;

    // Инструкции к миграции на вторую версию
    if (oldVersion < 2) {
        // Создаём хранилище для заметок
        db.createObjectStore('notes', {
            keyPath: 'id',
            autoIncrement: true
        });
    }
}
```

Добавление объекта

```
// Указываем на какие Store распространяется транзакция
const transaction = db.transaction(['notes'], 'readwrite');

// В рамках транзакции получаем ссылку на Store
const store = transaction.objectStore('notes');

// Добавляем заметку
const request = store.add({
    id: 'films'
    name: 'films'
});

request.onerror = err => console.error;

transaction.abort();
```

Получение объекта

```
// Указываем на какие Store распространяется транзакция
const transaction = db.transaction(['notes'], 'readonly');

// В рамках транзакции получаем ссылку на Store
const store = transaction.objectStore('notes')

// Получаем используя значения ключа (id)
const request = store.get('films')

request.onsuccess = note => console.log
```

Получение всех объектов

```
const transaction = db.transaction(['notes'], 'readonly');
const store = transaction.objectStore('notes')

const requestCursor = store.openCursor();

requestCursor.onsuccess = event => {
    const cursor = event.target.result;

    // Выводим данные текущего объекта
    console.log(cursor.key, cursor.value);

    // Переходим к следующему
    cursor.continue();
};

};
```

Индексы

```
requestDb.onupgradeneeded = event => {
    const db = event.target.result;
    const store = db.createObjectStore('notes', {
        keyPath: 'id',
        autoIncrement: true
    });

    store.createIndex(
        'name', // Названия
        'name', // Ключ
        { unique: false } // Параметры
    );
};
```

ПОИСК ПО УСЛОВИЮ

```
const transaction = db.transaction(['notes'], 'readonly');
const store = transaction.objectStore('notes')

const requestCursor = store
  .index('name')
  .openCursor(IDBKeyRange.only(['films', true]));

// ...
```

<i>Firefox 43</i>			
In-memory	1	12	152
LocalStorage	19	177	1950
IndexedDB	114	823	8849

```
const db = new Dexie('MyDatabase');
```

```
db
```

```
  .version(1)
  .stores({
    notes: 'name'
 });
```

```
db
```

```
  .open()
  .catch(error => console.error);
```

```
db  
  .notes  
  
  .where('name')  
  .equals(['films'])  
  .each(note => {  
    console.log(note.name);  
});
```

Using IndexedDB