عنوان: **آموزش PouchDB : قسمت اول (معرفی)**
نویسنده: محمد رضا صفری
تاریخ: ۱۳۹۳/۰۱/۲۵ ۱۷:۴۰
آدرس: www.dotnettips.info
گروه‌ها: NoSQL, Database, PouchDB, IndexedDB

آموزش PouchDB : معرفی

هدف این مقاله بر این است که شما را با دیتابیس PouchDB آشنا سازد .

در مطلب اول هدف فقط آشنایی و نحوه نصب PouchDB قرار خوهد داشت و در مطالب بعدی نحوه آشنایی با نحوه کدنویسی و استفاده به صورت آفلاین یا آنلاین بررسی خواهد شد .

فهرست مطالب :
بخش اول : معرفی PouchDB
شروع به کار با PouchDB
نحوه استفاده از API ها
سوالات متداول در مورد PouchDB
خطاهای احتمالی
پروژه‌ها و پلاگین های PouchDB

PouchDB یک دیتابیس NoSQL می‌باشد که به وسیله Javascript نوشته شده و هدف آن این است که برنامه نویس‌ها بتوانند برنامه‌هایی را توسعه و ارائه کنند که بتواند هم به صورت آفلاین و هم آنلاین سرویس دهی داشته باشند.
برنامه اطلاعات خودش را به صورت آفلاین ذخیره می‌کند و کاربر می‌تواند زمانیکه به اینترنت متصل نیست، از آنها استفاده کند. اما به محض اتصال به اینترنت، دیتابیس خودش را با دیتابیس آنلاین همگام (Sync) می‌کند. اینجاست که قدرت اصلی PouchDB مشخص می‌شود. بزرگترین برتری PouchDB همین است. دیتابیسی است که به صورت توکار قابلیت‌های همگام سازی را دارا می‌باشد و به صورت اتوماتیک این کار را انجام می‌دهد. PouchDB یک پروژه‌ی اوپن سورس است که توسط این افراد به روز می‌شود. البته باید گفت که PouchDB از CouchDB الهام گرفته شده است. اگر شما هم قصد همکاری در این پروژه را دارید بهتر است که راهنمای همکاری را مطالعه کنید .

پشتیبانی مرورگرها

PouchDB پیش زمینه‌های مختلفی دارد که به آن این امکان را می‌دهد تا روی همه مرورگرها و صد البته روی NodeJs کار کند. از IndexedDB بر روی Firefox/Chrome/Opera/IE و WebSql بر روی Safari و همچنین LevelDB بر روی NodeJs استفاده می‌کند.
در حال حاظر PouchDB بر روی مرورگرهای زیر تست شده است:
فایرفاکس 12 و بالاتر
گوگل کروم 19 و بالاتر
اپرا 12 و بالاتر
سافاری 5 و بالاتر
اینترنت اکسپلورر 10 و بالاتر
NodeJs 0.10 و بالاتر
و به صورت شگفت انگیزی در Apache Cordova

برای اطلاعات بیشتر در مورد مرورگرهایی که IndexdDB و WebSql را پشتیبانی می‌کنند به لینک‌های زیر مراجعه کنید:
Can I use IndexedDB

Can I use Web SQL Database

نکته : در صورتی که برنامه شما نیاز دارد تا از اینترنت اکسپلورر نسخه پایینتر از 10 استفاده کند می‌توانید از دیتابیس‌های آنلاین استفاده کنید، که البته دیگر قابلیت استفاده آفلاین را نخواهد داشت.

وضعیت فعلی PouchDB

PouchDB برای مرورگر، فعلا در وضعیت بتا به سر می‌برد و به صورت فعالی در حال گذراندن تست هایی می‌باشد تا باگ‌های آن برطرف شود و به صورت پایدار (Stable) ارائه گردد. البته فقط ممکن است که شما باگی را در قسمت Apiها پیدا کنید که البته Apiها هم در حال حاضر پایدار هستند و گزارشی مبنی بر باگ در آن‌ها موجود نیست. اگر هم باگی پیدا بشود شما می‌توانید PouchDB را بدون ریسک از دست رفتن اطلاعات آپگرید کنید.

PouchDB برای NodeJs فعلا در وضعیت آلفا است و آپگرید کردن ممکن است به اطلاعات شما آسیب بزند. البته با آپدیت به صورت دستی خطری شما را تهدید نخواهد کرد .

نحوه‌ی نصب PouchDB

PouchDB به صورت یک کتابخانه‌ی کوچک و جمع و جور طراحی شده است تا بتواند همه نیازها را برطرف و روی همه نوع Device اعم از موبایل، تبلت، مرورگر و کلا هر چیزی که جاوا اسکریپت را ساپورت می‌کند کار خود را به خوبی انجام بدهد.

برای استفاده از PouchDB میبایست این فایل را با حجم فوق العاده 97 کیلوبایت دانلود کنید و آن را به یک صفحه وب اضافه کنید :

```
<script src="pouchdb-2.1.0.min.js"></script>
```

آخرین نسخه و بهترین نسخه : pouchdb-2.1.0.min.js
برای اطلاع از آخرین آپدیتها و نسخه‌ها به این صفحه در گیت هاب مراجعه کنید .
برای کسانی هم که از NodeJS استفاده می‌کنند نحوه نصب به این صورت است :

```
$ npm install pouchdb
```

نظرات خوانندگان

نویسنده: سعیدزمان
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۱:۲۰

سلام مهندس مطلب بسیار جالبی بود فقط من یک سوال برام پیش اومده که این اطلاعات که میخواد در حالت افلاین استفاده بشه کجا قرار میگیره ؟ ایا امنیت داده‌های حساس رو پایین نمی‌یاره ؟
با تشکر

نویسنده: محمد رضا صفری
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۵:۱۵

پیشنهاد می‌کنم این اسلاید هارو ببینید : http://www.slideshare.net/wurbanski/nosql-no-security
http://www.slideshare.net/gavinholt/nosql-no-security-16514872
واقعا مفیده و خیلی از سوالات شمارو پاسخ میده .
موفق باشید ./

عنوان: **WebStorage**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۰۸ ۱۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Cookie, IndexedDB, Webstorage, HTML 5

webstorage تقریبا فناوری جدیدی است که برای نگهداری ثابت داده‌ها بر روی سیستم کاربر استفاده می‌شود. webstorage مزایای زیادی برای برنامه‌های تحت وب دارد. برای مثال با استفاده از آن میتوان فعالیت‌های کاربر را رصد کرد، بدون اینکه کد و دیتابیس سمت سرور را دخالت دهیم. حتی اگر سیستم کاربر آفلاین هم بشود برنامه می‌تواند همچنان به فعالیتش ادامه دهد. در این مقاله به مزایای این روش می‌پردازیم.

WebStorage در برابر کوکی ها

یکی از روش‌های سنتی ذخیره اطلاعات در سیستم کاربر، کوکی‌ها در بستر Http هستند. تفاوت‌های زیادی بین این دو وجود دارد که تعدادی از آن‌ها را در زیر بررسی می‌کنیم:

مکانیزم ذخیره سازی:

کوکی‌ها داده‌های ساخت یافته‌ای هستند که از وب سرور به سمت مرورگر کاربر به عنوان پاسخی در ازای درخواستی ارسال می‌شوند. درخواست و پاسخ کوکی‌ها شامل بخش هدر بوده که اطلاعات آن باعث شناسایی کوکی برای مدیریت و شناسایی آن‌ها می‌گردد تا هر موقع درخواستی صورت بگیرد، به سمت سرور برگشت خواهد خورد.

به طور خلاصه اینکه کوکی‌ها توسط درخواست‌ها و پاسخ‌ها ایجاد یا به روز می‌شوند. در نتیجه داده‌ها چه تغییر کرده باشند چه تغییر نکرده باشند، همیشه بخشی از هدر Http هستند. در سوی دیگر webstorage به طور کامل به صورت کلاینتی پیاده سازی گشته است و وب سرور را درگیر کار خودشان نمی‌کنند و کارایی بهتری را ارائه می‌دهند. از آنجا که همه چیز در خود سیستم کاربر اتفاق می‌افتد، در صورت از دست دادن کانکشن شبکه، کاربر می‌تواند همچنان به فعالیت‌های به روزرسانی و تغییر ادامه دهد.

چند نسخه از مرورگر

کاربری که با وب سایت مدنظر کار میکند میتواند توسط چند مرورگر مختلف یا چند تب و پنجره مختلف به طور همزمان کار کند و اطلاعات آن در همه‌ی مرورگرها و دیگر پنجره‌های آن مرورگر قابل دسترس می‌باشد.

محدودیت حجمی

محدودیت کوکی‌ها و webstorage از نظر حجم ذخیره سازی بین مروگرهای مختلف، متفاوت است. ولی در حالت کلی در بیشتر مرورگرها محدودیت حجم 4 کیلوبایت برای کوکی‌ها موجود است. (این ابزار می تواند نهایت حجمی را که که مرورگر شما از کوکی پشتیبانی می‌کند، نشان دهد.)

در مورد webstorage استاندارد W3C محدودیتی اعلام نکرده است و تصمیم گیری بر سر این موضوع را به سازندگان مرورگرها واگذار کرده است. ولی در حالت کلی حجم بیشتری از کوکی را ذخیره میکند و عموما تا 5 مگابایت توانایی ذخیره سازی وجود دارد. بدین ترتیب حجم آن 124,527% بیشتر از کوکی‌ها است. (این ابزار می‌تواند نهایت حجمی را که مرورگر شما از webstorage پشتیبانی می‌کند، ببینید).

انواع Webstorage

دو نوع متد ذخیره سازی در webstorage داریم:

session storage

local storage

Web Storage type	Lifetime of stored data	Data structure	Data type
sessionStorage	Session only	Key/value pairs	String
localStorage	Persistent	Key/value pairs	String

SessionStorage

داده‌هایی که بدین صورت ذخیره می‌شوند تنها تا زمانی دوام آورده و پایدار هستند که session مرورگر فعال است؛ یعنی تا زمانیکه کاربر در سایت فعلی حضور دارد.
استفاده از این روش برای ذخیره سازی‌های موقت عالی است. برای نمونه مقادیر ورودی فرمی که کاربر در حال کار با آن است، میتواند به طور موقت ذخیره شوند تا زمانی که کاربر بتواند تمامی مراحل را طی کرده، بدون اینکه ارجاعی به دیتابیس سمت سرور داشته باشد. همچنین ذخیره موقت داده‌ها می‌تواند به کاربر کمک کند تا در صورت rereshهای ناگهانی یا بسته شدن‌های ناگهانی مرورگرها، نیازی به ورود مجدد داده‌ها نداشته باشد.

LocalStorage

در این روش داده‌ها با از دست رفتن session مرورگر جاری از بین نمی‌رود و برای بازدیدهای آتی کاربر از سایت، داده‌ها همچنان در دسترس هستند. **پشتیبانی مرورگرها**
وب سایت caniuse گزارش می‌دهد که اکثر مرورگرها پشتیبانی خوبی از webstorage دارند.

Browser	Version
Internet Explorer	**IE 8** and above
Mozilla Firefox	**Firefox 3.5** and above
Google Chrome	**Chrome 4** and above
Apple Safari	**Safari 4** and above
Opera	**Opera 11.5** and above

با اینکه توصیه نامه W3C از پایان کار پیاده سازی این قابلیت خبر میدهد ولی در حال حاضر که این مقاله تدوین شده است هنوز نهایی اعلام نشده است. برای پشتیبانی مرورگرهای قدیمی از webstorage می‌توان از فایل جاوااسکریپتی Store.js کمک گرفت.

مفاهیم امنیتی و محافظت از داده ها

محدودیت‌های حمایتی و حفاظتی webstorage دقیقا همانند کوکی هاست. به این معنی که وب سایت‌های دیگر توانایی اتصال به webstorage سایت دیگری را ندارند. البته این مورد ممکن است برای وب سایت هایی که بر ساب دومین تکیه کرده‌اند ایجاد مشکل کند. برای حل این مسائل می‌توانید از کتابخانه‌های سورس بازی چون Cross Storage که توسط Zendesk ارائه شده است، استفاده کرد.

همانند هر مکانیزم ذخیره سازی سمت کلاینت، مواردی توصیه میگردد که رعایت آن‌ها از لحاظ امنیتی پر اهمیت است. به عنوان نمونه ذخیره‌ی اطلاعات شخصی و موارد حساس توصیه نمی‌گردد؛ چرا که احتمال دسترسی آسان نفوذگران به داده‌های محلی و خواندن آن‌ها وجود دارد.

Data Integrity یا یکپارچگی داده‌ها نیز در نظر گرفته شده است. باید حفاظتی در برابر عدم موفقیت ذخیره سازی داده‌ها نیز وجود داشته باشد. این عدم موفقیت‌ها میتواند به دلایل زیر رخ دهد:
اگر کاربر قابلیت webstorage را غیرفعال کرده باشد.

اگر فضایی برای کاربر باقی نمانده باشد.

با محدودیت حجمی webstorage مواجه شده است.

با مواجه شدن با خطاها یک استثنا صادر می‌شود که می‌توانید آن را دریافت و کنترلی را روی برنامه تحت وب داشته باشید. یک نمونه استثنا QuotaExceededError

IndexedDB

یکی از فرایندهای ذخیره سازی داده‌ها که همان مزایای webstorage را ارائه میدهد indexed Database API است. این قابلیت از HTML 5 اضافه شده است و قسمتی از مشخصات webstorage شناخته نمی‌شود. برای همین مستنداتی در حوزه‌ی webstorage برای آن پیدا نخواهید کرد ولی قابلیت‌هایی فراتر از webstorage دارد.

این قابلیت پیچیدگی بیشتری را نسبت به خود webstorage ایجاد می‌کند، ولی فرصت‌های بسیاری را برای ذخیره سازی داده‌هایی با معماری‌های پیچیده‌تر و رابطه‌ها را می‌دهد. با استفاده از IndexedDB داده‌ها به شکل دیتابیس‌های سمت سرور RDMS ذخیره می‌شوند و این قابلیت را دارید که به سمت آن کوئری هایی مشابه بانک‌های اطلاعاتی سمت سرور را ارسال کنید.

در قسمت آتی نحوه کدنویسی آن را فرا خواهیم گرفت.

نظرات خوانندگان

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۱۴ ۲:۵۹

آیا روش‌های ذکر شده (Session & Local Storage) برای طراحی یک سبد خرید (در یک پروژه فروشگاه اینترنتی) مناسب هستند؟ اگر مناسب هستند، بهتر از Session معمولی کار میکنند؟

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۱۴ ۳:۷

اینجا را بخوانید

عنوان: **WebStorage: قسمت دوم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۰۹ ۱:۰
آدرس: www.dotnettips.info
گروه‌ها: Cookie, IndexedDB, Webstorage, HTML 5

در این مقاله قصد داریم نحوه‌ی کدنویسی webstorage را با کتابخانه‌هایی که در مقاله قبل معرفی کردیم بررسی کنیم.
ابتدا روش ذخیره سازی و بازیابی متداول آن را بررسی میکنیم که تنها توسط دو تابع صورت می‌گیرد. مطلب زیر برگرفته از w3Schools است:
دسترسی به شیء webstorage به صورت زیر امکان پذیر است:

```
window.localStorage
window.sessionStorage
```

ولی بهتر است قبل از ذخیره و بازیابی، از پشتیبانی مرورگر از webstorage اطمینان حاصل نمایید:

```
if(typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

برای ذخیره سازی و سپس خواندن به شکل زیر عمل می‌کنیم:

```
localStorage.setItem("lastname", "Smith");

//========================
localStorage.getItem("lastname");
```

خواندن می‌تواند حتی به شکل زیر هم صورت بگیرد:

```
var a=localStorage.lastname;
```

استفاده از store.js برای مرورگرهایی که از webstorage پشتیبانی نمی‌کنند به شکل زیر است:

```
//ذخیره مقدار
store.set('username', 'marcus')

//بازیابی مقدار
store.get('username')

//حذف مقدار
store.remove('username')

//حذف تمامی مقادیر ذخیره شده
store.clear()

//ذخیره ساختار
store.set('user', { name: 'marcus', likes: 'javascript' })

//بازیابی ساختار به شکل قبلی
var user = store.get('user')
alert(user.name + ' likes ' + user.likes)

//تغییر مستقیم مقدار قبلی
store.getAll().user.name == 'marcus'

//بازخوانی تمام مقادیر ذخیر شده توسط یک حلقه
store.forEach(function(key, val) {
    console.log(key, '==', val)
})
```

همچنین بهتر هست از یک فلگ برای بررسی فعال بودن storage استفاده نمایید. به این دلیل که گاهی کاربرها از پنجره‌های

private استفاده می‌کنند که ردگیری اطلاعات آن ممکن نیست و موجب خطا می‌شود.

```
<script src="store.min.js"></script>
<script>
    init()
    function init() {
        if (!store.enabled) {
            alert('Local storage is not supported by your browser. Please disable "Private Mode", or
upgrade to a modern browser.')
            return
        }
        var user = store.get('user')
        // ... and so on ...
    }
</script>
```

در صورتیکه بخشی از داده‌ها را توسط localstorage ذخیره نمایید و بخواهید از طریق storage به آن دسترسی داشته باشید، خروجی string خواهد بود؛ صرف نظر از اینکه شما عدد، شیء یا آرایه‌ای را ذخیره کرده‌اید.
در صورتیکه ساختار JSON را ذخیره کرده باشید، می‌توانید رشته برگردانده شده را با json.stringify و json.parse بازیابی و به روز رسانی کنید.
در حالت cross browser تهیه‌ی یک sessionStorage امکان پذیر نیست. ولی می‌توان به روش ذیل و تعیین یک زمان انقضاء آن را محدود کرد:

```
var storeWithExpiration = {
// دریافت کلید و مقدار و زمان انقضا به میلی ثانیه
    set: function(key, val, exp) {
//ایجاد زمان فعلی جهت ثبت تاریخ ایجاد
        store.set(key, { val:val, exp:exp, time:new Date().getTime() })
    },
    get: function(key) {
        var info = store.get(key)
//در صورتی که کلید داده شده مقداری نداشته باشد نال را بر میگردانیم
        if (!info) { return null }
//تاریخ فعلی را منهای تاریخ ثبت شده کرده و در صورتی که
//از مقدار میلی ثاینه بیشتر باشد یعنی منقضی شده و نال بر میگرداند
        if (new Date().getTime() - info.time > info.exp) { return null }
        return info.val
    }
}

// استفاده عملی از کد بالا
// استفاده از تایمر جهت نمایش واکشی داده‌ها قبل از نقضا و بعد از انقضا
storeWithExpiration.set('foo', 'bar', 1000)
setTimeout(function() { console.log(storeWithExpiration.get('foo')) }, 500) // -> "bar"
setTimeout(function() { console.log(storeWithExpiration.get('foo')) }, 1500) // -> null
```

مورد بعدی استفاده از سورس cross-storage است. اگر به یاد داشته باشید گفتیم یکی از احتمالاتی که برای ما ایجاد مشکل می‌کند، ساب دومین هاست که ممکن است دسترسی ما به یک webstorage را از ساب دومین دیگر از ما بگیرد.
این کتابخانه به دو جز تقسیم شده است یکی هاب Hub و دیگری Client .

ابتدا نیاز است که هاب را آماده سازی و با ارائه یک الگو از آدرس وب، مجوز عملیات را دریافت کنیم. در صورتیکه این مرحله به فراموشی سپرده شود، انجام هر نوع عمل روی دیتاها در نظر گرفته نخواهد شد.

```
CrossStorageHub.init([
  {origin: /\.example.com$/,           allow: ['get']},
  {origin: /:\/\/(www\.)?example.com$/, allow: ['get', 'set', 'del']}
]);
```

حرف $ در انتهای عبارت باعث مشود که دامنه‌ها با دقت بیشتری در Regex بررسی شوند و دامنه زیر را معتبر اعلام کند:

```
valid.example.com
```

ولی دامنه زیر را نامعتبر اعلام می‌کند:

```
invalid.example.com.malicious.com
```

همچنین می‌توانید تنظیماتی را جهت هدرهای CSP و CORS، نیز اعمال نمایید:

```
{
  'Access-Control-Allow-Origin':  '*',
  'Access-Control-Allow-Methods': 'GET,PUT,POST,DELETE',
  'Access-Control-Allow-Headers': 'X-Requested-With',
  'Content-Security-Policy':      "default-src 'unsafe-inline' *",
  'X-Content-Security-Policy':    "default-src 'unsafe-inline' *",
  'X-WebKit-CSP':                 "default-src 'unsafe-inline' *",
}
```

پس کار را بدین صورت آغاز می‌کنیم، یک فایل به نام hub.htm درست کنید و هاب را آماده سازید: **hub.htm**

```
<script type="text/javascript" src="~/Scripts/cross-storage/hub.js"></script>
<script>
 CrossStorageHub.init([
      {origin: /.*localhost:300\d$/, allow: ['get', 'set', 'del']}
    ]);
  </script>
```

کد بالا فقط درخواست‌های هاست لوکال را از پورتی که ابتدای آن با 300 آغاز می‌شود، پاسخ می‌دهد و مابقی درخواست‌ها را رد می‌کند. متدهای ایجاد، ویرایش و حذف برای این آدرس معتبر اعلام شده است.
در فایل دیگر که کلاینت شناخته می‌شود باید فایل hub معرفی شود تا تنظیمات هاب خوانده شود:

```
var storage = new CrossStorageClient('http://localhost:3000/example/hub.html');

var setKeys = function () {

return storage.set('key1', 'foo').then(function() {
return storage.set('key2', 'bar');
      });
};
```

در خط اول، فایل هاب معرفی شده و تنظیمات روی این صفحه اعمال می‌شود. سپس در خطوط بعدی داده‌ها ذخیره می‌شوند. از آنجا که با هر یکبار ذخیره، return صورت می‌گیرد و تنها اجازه‌ی ورود یک داده را داریم، برای حل این مشکل متد then پیاده سازی شده است. متغیر setKeys شامل یک آرایه از کلیدها خواهد بود.
نحوه‌ی ذخیره سازی بدین شکل هم طبق مستندات صحیح است:

```
storage.onConnect().then(function() {

  return storage.set('key', {foo: 'bar'});

}).then(function() {

  return storage.set('expiringKey', 'foobar', 10000);

});
```

در کد بالا ابتدا یک داده دائم ذخیره شده است و در کد بعد یک داده موقت که بعد از 10 ثانیه اعتبار خود را از دست می‌دهد.
برای خواندن داده‌های ذخیره شده به نحوه زیر عمل می‌کنیم:

```
storage.onConnect().then(function() {
  return storage.get('key1');
}).then(function(res) {
  return storage.get('key1', 'key2', 'key3');
}).then(function(res) {
  // ...
});
```

کد بالا نحوه‌ی خواندن مقادیر را به شکل‌های مختلفی نشان میدهد و مقدار بازگشتی آن‌ها یک آرایه از مقادیر است؛ مگر اینکه تنها یک مقدار برگشت داده شود. مقدار بازگشتی در تابع بعدی به عنوان یک آرگومان در دسترس است. در صورتی که خطایی رخ دهد، قابلیت هندل آن نیز وجود دارد:

```
storage.onConnect()
    .then(function() {
      return storage.get('key1', 'key2');
    })

.then(function(res) {
      console.log(res); // ['foo', 'bar']
    })['catch'](function(err) {
      console.log(err);
    });
```

برای باقی مسائل چون به دست آوردن لیست کلیدهای ذخیره شده، حذف کلیدهای مشخص شده، پاکسازی کامل داده‌ها و ... به مستندات رجوع کنید.

در اینجا جهت سازگاری با مرورگرهای قدیمی خط زیر را به صفحه اضافه کنید:

```
<script src="https://s3.amazonaws.com/es6-promises/promise-1.0.0.min.js"></script>
```

ذخیره‌ی اطلاعات به شکل یونیکد، فضایی دو برابر کدهای اسکی میبرد و با توجه به محدود بودن حجم webstorage به 5 مگابایت ممکن است با کمبود فضا مواجه شوید. در صورتیکه قصد فشرده سازی اطلاعات را دارید می‌توانید از کتابخانه lz-string استفاده کنید. ولی توجه به این نکته ضروری است که در صورت نیاز، عمل فشرده سازی را انجام دهید و همینطوری از آن استفاده نکنید.

IndexedDB API

آخرین موردی که بررسی می‌شود استفاده از IndexedDB API است که با استفاده از آن میتوان با webstorage همانند یک دیتابیس رفتار کرد و به سمت آن کوئری ارسال کرد.

```
var request = indexedDB.open("library");

request.onupgradeneeded = function() {
  // The database did not previously exist, so create object stores and indexes.
  var db = request.result;
  var store = db.createObjectStore("books", {keyPath: "isbn"});
  var titleIndex = store.createIndex("by_title", "title", {unique: true});
  var authorIndex = store.createIndex("by_author", "author");

  // Populate with initial data.
  store.put({title: "Quarry Memories", author: "Fred", isbn: 123456});
  store.put({title: "Water Buffaloes", author: "Fred", isbn: 234567});
  store.put({title: "Bedrock Nights", author: "Barney", isbn: 345678});
};

request.onsuccess = function() {
  db = request.result;
};
```

کد بالا ابتدا به دیتابیس library متصل می‌شود و اگر وجود نداشته باشد، آن را می‌سازد. رویداد onupgradeneeded برای اولین بار اجرا شده و در آن می‌توانید به ایجاد جداول و اضافه کردن داده‌های اولیه بپردازید؛ یا اینکه از آن جهت به ارتقاء ورژن دیتابیس استفاده کنید. خصوصیت result، دیتابیس باز شده یا ایجاد شده را باز می‌گرداند. در خط بعدی جدولی با کلید کد ISBN کتاب تعریف شده است. در ادامه هم دو ستون اندیس شده برای عنوان کتاب و نویسنده معرفی شده است که عنوان کتاب را یکتا و بدون تکرار در نظر گرفته است. سپس در جدولی که متغیر store به آن اشاره می‌کند، با استفاده از متد put، رکوردها داخل آن درج می‌شوند. در صورتیکه کار با موفقیت انجام شود رویداد onSuccess فراخوانی می‌گردد.

برای انجام عملیات خواندن و نوشتن باید از تراکنش‌ها استفاده کرد:

```
var tx = db.transaction("books", "readwrite");
var store = tx.objectStore("books");

store.put({title: "Quarry Memories", author: "Fred", isbn: 123456});
store.put({title: "Water Buffaloes", author: "Fred", isbn: 234567});
```

```
store.put({title: "Bedrock Nights", author: "Barney", isbn: 345678});

tx.oncomplete = function() {
  // All requests have succeeded and the transaction has committed.
};
```

در خط اول ابتدا یک خط تراکنشی بین ما و جدول books با مجوز خواندن و نوشتن باز می‌شود و در خط بعدی جدول books را در اختیار می‌گیریم و همانند کد قبلی به درج داده‌ها می‌پردازیم. در صورتیکه عملیات با موفقیت به اتمام برسد، متغیر تراکنش رویدادی به نام oncomplete فراخوانی می‌گردد. در صورتیکه قصد دارید تنها مجوز خواندن داشته باشید، عبارت readonly را به کار ببرید.

```
var tx = db.transaction("books", "readonly");
var store = tx.objectStore("books");
var index = store.index("by_author");

var request = index.openCursor(IDBKeyRange.only("Fred"));
request.onsuccess = function() {
  var cursor = request.result;
  if (cursor) {
    // Called for each matching record.
    report(cursor.value.isbn, cursor.value.title, cursor.value.author);
    cursor.continue();
  } else {
    // No more matching records.
    report(null);
  }
};
```

در دو خط اول مثل قبل، تراکنش را دریافت می‌کنیم و از آنجا که می‌خواهیم داده را بر اساس نام نویسنده واکشی کنیم، ستون اندیس شده نام نویسنده را دریافت کرده و با استفاده از متد opencursor درخواست خود را مبنی بر واکشی رکوردهایی که نام نویسنده **fred** است، ارسال میداریم. در صورتیکه عملیات با موفقیت انجام گردد و خطایی دریافت نکنیم رویداد onsuccess فراخوانی می‌گردد. در این رویداد با دو حالت برخورد خواهیم داشت؛ یا داده‌ها یافت می‌شوند و رکوردها برگشت داده می‌شوند یا هیچ رکوردی یافت نشده و مقدار نال برگشت خواهد خورد. با استفاده از cursor.continue می‌توان داده‌ها را به ترتیب واکشی کرده و مقادیر رکورد را با استفاده خصوصیت value به سمت تابع report ارسال کرد.

کدهای بالا همه در مستندات معرفی شده وجود دارند و ما پیشتر توضیح ابتدایی در مورد آن دادیم و برای کسب اطلاعات بیشتر می‌توانید به همان مستندات معرفی شده رجوع کنید. برای idexedDB هم می‌توانید از این منابع + + + استفاده کنید که خود w3c منبع فوق العاده‌تری است.