IndexedDB 基本使用

Web Storage 使用简单字符串键值对在本地存储数据，方便灵活，但是对于大量结构化数据存储力不从心，IndexedDB 是为了能够在客户端存储大量的结构化数据，并且使用索引高效检索的 API。

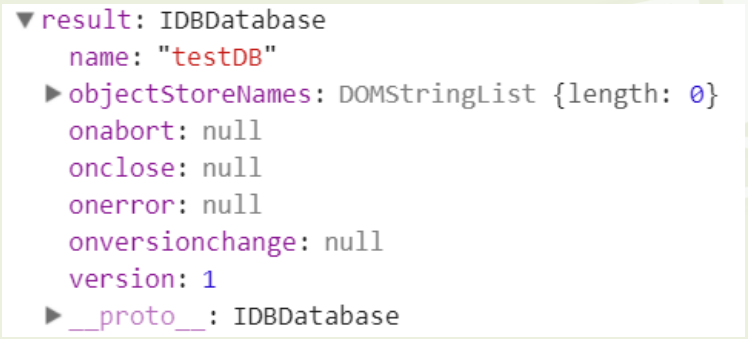
异步 API

在 IndexedDB 大部分操作并不是我们常用的调用方法，返回结果的模式，而是请求——响应的模式，比如打开数据库的操作

```
var request=window.indexedDB.open('testDB');
```

这条指令并不会返回一个 DB 对象的句柄，得到的是一个 IDBOpenDBRequest 对象，而我们希望得到的 DB 对象在其 result 属性中，

这条指令请求的响应是一个 IDBDatabase 对象，这就是 IndexedDB 对象，



除了 result，IDBOpenDBRequest 接口定义了几个重要属性

- onerror：请求失败的回调函数句柄
- onsuccess:请求成功的回调函数句柄
- onupgradeneeded:请求数据库版本变化句柄

所谓异步 API 是指并不是这条指令执行完毕，我们就可以使用 request.result 来获取 indexedDB 对象了，语句执行完并不代表已经获取到了对象，所以一般在其回调函数中处理。

创建数据库

刚才的语句已经展示了如何打开一个 indexedDB 数据库，调用 indexedDB.open 方法就可以创建或者打开一个 indexedDB。看一个完整的处理

```
 var dbName = 'testDB',
     version = 1;
function openDB (name) {
          var request=window.indexedDB.open(name);
          request.onerror=function(e){
              console.log(e.currentTarget.error.message);
          };
          request.onsuccess=function(){
              db=request.result;
              console.log('open success');
          };
       }
openDB(dbName);
```

把 request 获取的 DB 对象赋值给了 db，这样就可以使用 db 来访问创建的 indexedDB 了。

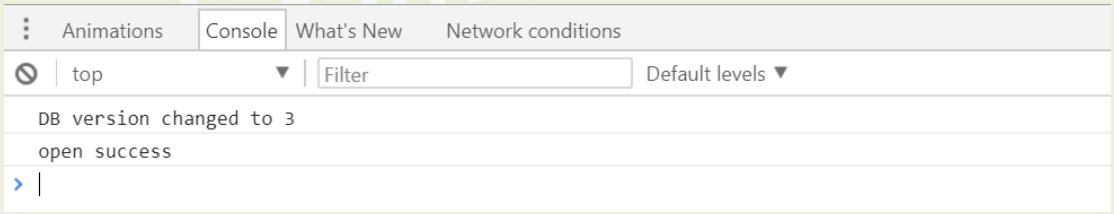
version

除了 onerror 和 onsuccess，IDBOpenDBRequest 还有一个类似回调函数句柄——onupgradeneeded。**这个句柄在我们请求打开的数据库的版本号和已经存在的数据库版本号不一致的时候调用。**

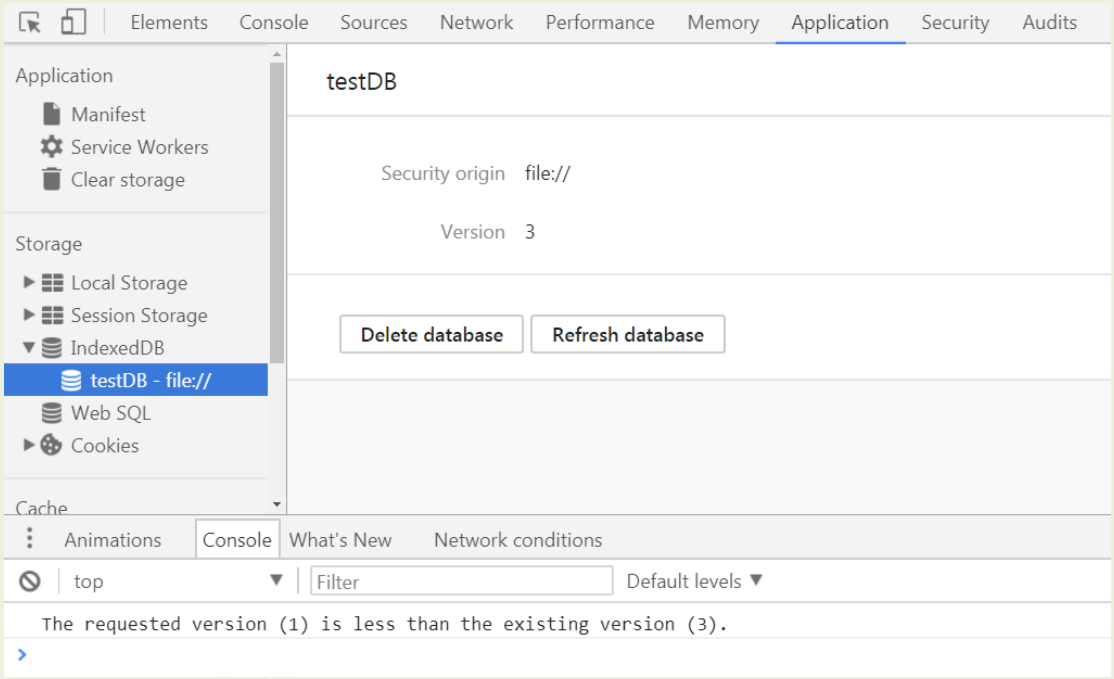
indexedDB.open()方法还有第二个可选参数，数据库版本号，数据库创建的时候默认版本号为 1，当我们传入的版本号和数据库当前版本号不一致的时候 onupgradeneeded 就会被调用，**当然不能试图打开比当前数据库版本低的 version，否则调用的就是 onerror**，修改一下刚才例子

```
 var dbName = 'testDB',
     version = 3;
function openDB (name,version) {
          var version=version || 1;
          //打开或创建数据库
          var request=window.indexedDB.open(name,version);
          request.onerror=function(e){
              console.log(e.currentTarget.error.message);
          };
          request.onsuccess=function(e){
              db=request.result; //这里才是 indexedDB 对象
              console.log('open success');
          };
          // 注意：只能请求>=当前数据库版本号的版本
          // 大于当前版本号，则触发 onupgradeneeded,
          //小于当前版本号，则触发 onerror
          request.onupgradeneeded=function(){
              console.log('DB version changed to '+version);
          };
       }
       openDB(dbName,version);
```

由于刚才已经创建了版本为 1 的数据库，打开版本为 3 的时候，会在控制台输出：DB version changed to 3。



当将版本修改为 1 后，在执行代码，报错：



关闭与删除数据库

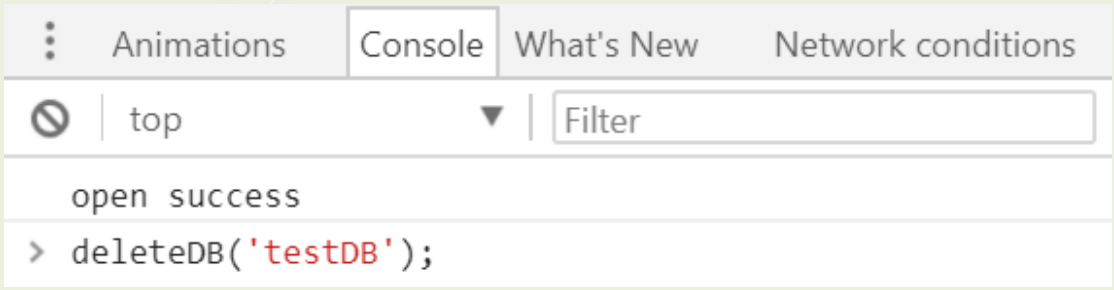
关闭数据库可以直接调用数据库对象的 close 方法

```
function closeDB(db){
            db.close();
        }
```

删除数据库使用 indexedDB 对象的 deleteDatabase 方法

```
function deleteDB(name){
            indexedDB.deleteDatabase(name);
        }
```

由于异步 API 愿意，不能保证能够在 closeDB 方法调用前获取 db 对象（实际上获取 db 对象也比执行一条语句慢得多），所以建议在执行完成该代码后，在浏览器的控制中调用“关闭和删除”数据库的方法，而不是在代码中直接调该方法。



object store

有了数据库后，自然希望创建一个表用来存储数据，但 indexedDB 中没有表的概念，而是 objectStore，一个数据库中可以包含多个 objectStore，objectStore 是一个灵活的数据结构，可以存放多种类型数据。也就是说一个 objectStore 相当于一张表，里面存储的每条数据和一个键相关联。

可以使用每条记录中的某个指定字段作为键值（keyPath），也可以使用自动生成的递增数字作为键值（keyGenerator），也可以不指定。选择键的类型不同，objectStore 可以存储的数据结构也有差异

键类型	存储数据
不使用	任意值，但是没添加一条数据的时候需要指定键参数
keyPath	Javascript 对象，对象必须有一属性作为键值
keyGenerator	任意值
都使用	Javascript 对象，如果对象中有 keyPath 指定的属性则不生成新的键值，如果没有自动生成递增键值，填充 keyPath 指定属性

事务

在对新数据库做任何事情之前，需要开始一个事务。事务要求指定 object store。

事务具有三种模式

1. 只读：read，不能修改数据库数据，可以并发执行
2. 读写：readwrite，可以进行读写操作
3. 版本变更：verionchange

```
Var transaction = db.transaction(os1, 'readwrite');
//打开一个事务
var objectStore=transaction.objectStore('os1');
//获取 os1 object store
```

给 object store 添加数据

调用数据库实例的 createObjectStore 方法可以创建 object store，方法有两个参数：store name 和键类型。调用 store 的 add 方法添加数据。有了上面知识，可以向 object store 内添加数据了

keyPath

因为对新数据的操作都需要在 transaction 中进行，而 transaction 又要求指定 object store，所以只能在创建数据库的时候初始化 object store 以供后面使用，这正是 onupgradeneeded 的一个重要作用，修改一下之前代码:

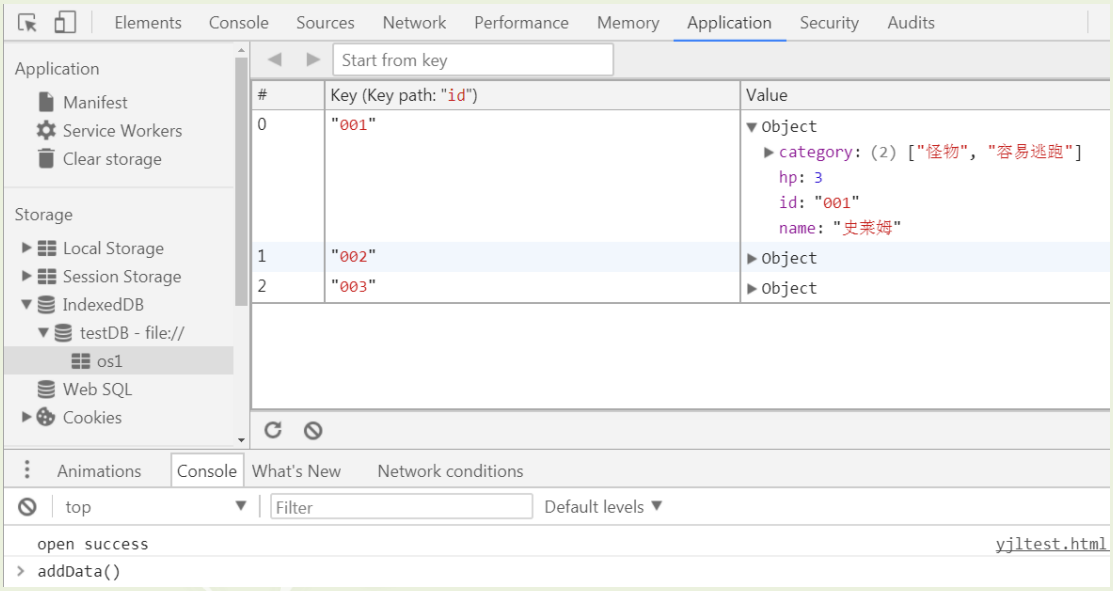
```
var dbName = 'testDB',
       version = 1,
       storeName = 'os1';
   function openDB(name, version) {
       var version = version || 1;
       var request = window.indexedDB.open(name, version);
       request.onerror = function(e) {
           console.log(e.currentTarget.error.message);
       };
       request.onsuccess = function(e) {
           db = request.result;
           console.log('open success');
       };
       request.onupgradeneeded = function(e) {
           var db = request.result;
           if (!db.objectStoreNames.contains('os1')) {
               db.createObjectStore('os1', { keyPath: "id" });
           }
           console.log('DB version changed to ' + version);
       };
   }
```

```
    openDB(dbName, version);
```

为其添加了一个名为 data 的 object store，准备一些数据以供添加，代码中添加如下代码：

```
var data = [{
        name: '史莱姆',
        id: '001',
        hp: 3,
        category: ['怪物', '容易逃跑']
    }, {
        name: '小蝙蝠',
        id: '002',
        hp: 5,
        category: ['怪物', '飞行']
    }, {
        name: '小恶魔',
        id: '003',
        hp: 9,
        category: ['怪物', '恶魔']
   }]
   function addData(db,storeName){
           var transaction=db.transaction(storeName,'readwrite');
           var store=transaction.objectStore(storeName);
           for(var i=0;i<data.length;i++){
              store.add(data[i]);
           }
       }
```

这样我们就在 os1 object store 里添加了三条记录，以 id 为键，运行代码，在控制台运行 addData()，在 chrome 控制台看看效果：

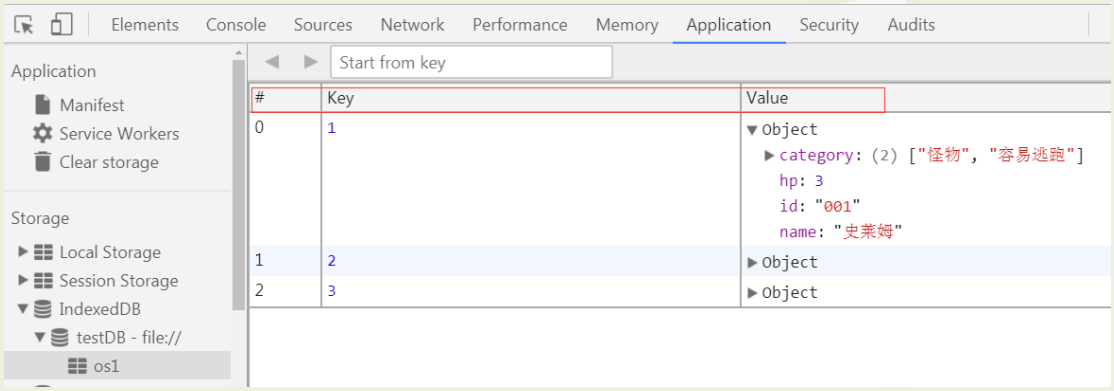


keyGenerate

```
function openDB(name, version) {
        var version = version || 1;
        var request = window.indexedDB.open(name, version);
        request.onerror = function(e) {
            console.log(e.currentTarget.error.message);
        };
        request.onsuccess = function(e) {
            db = request.result;
            console.log('open success');
        };
        request.onupgradeneeded = function(e) {
            var db = request.result;
            if (!db.objectStoreNames.contains('os1')) {
            db.createObjectStore('os1', {autoIncrement: true});
             // 更改为 autoIncrement: true
            }
            console.log('DB version changed to ' + version);
```

```
            };
        }
```

注意：因为更改 os1 键类型，在 onupgradeneeded 中，同时 db.objectStoreNames.contains('os1')已经存在，所以要运行成功，将数据库删除，然后在重新运行代码，并添加数据 addData()。



查找数据

调用 object store 的 get 方法通过键获取数据，**注意：使用 keyPath 做键为例**

```
function getDataByKey(value){
            var transaction=db.transaction(storeName,'readwrite');
            var store=transaction.objectStore(storeName);
            var request=store.get(value);
            request.onsuccess=function(){
                var os1result=request.result;
                console.log(os1result.name);
            };
}
```

删除数据及 object store

调用 object store 的 delete 方法根据键值删除记录

```
function deleteDataByKey(value){
            var transaction=db.transaction(storeName,'readwrite');
            var store=transaction.objectStore(storeName);
            store.delete(value);
        }
```

调用 object store 的 clear 方法可以清空 object store

```
function clearObjectStore(){
          var transaction=db.transaction(storeName,'readwrite');
          var store=transaction.objectStore(storeName);
          store.clear();
}
```

完整代码：

```
var dbName = 'testDB',
    version = 2,
    storeName = 'os1';

function openDB(name, version) {
    var version = version || 1;
    var request = window.indexedDB.open(name, version);
    request.onerror = function(e) {
        console.log(e.currentTarget.error.message);
    };
    request.onsuccess = function(e) {
        db = request.result;
        console.log('open success');
    };
    request.onupgradeneeded = function(e) {
        var db = request.result;
        if (!db.objectStoreNames.contains('os1')) {
            db.createObjectStore('os1', {keyPath:"id"});
        }
        console.log('DB version changed to ' + version);
    };
}
```

```
openDB(dbName, version);


var data = [{
    name: '史莱姆',
    id: '001',
    hp: 3,
    category: ['怪物', '容易逃跑']
}, {
    name: '小蝙蝠',
    id: '002',
    hp: 5,
    category: ['怪物', '飞行']
}, {
    name: '小恶魔',
    id: '003',
    hp: 9,
    category: ['怪物', '恶魔']
}]
function addData() {
    var transaction = db.transaction(storeName, 'readwrite');
    var store = transaction.objectStore(storeName);
    for (var i = 0; i < data.length; i++) {
        store.add(data[i]);
    }
}
function closeDB(db) {
    db.close();
}
function deleteDB(name) {
    indexedDB.deleteDatabase(name);
```

```
}
function getDataByKey(value){
        var transaction=db.transaction(storeName,'readwrite');
        var store=transaction.objectStore(storeName);
        var request=store.get(value);
        request.onsuccess=function(e){
            var os1result=request.result;
            console.log(os1result.name);
        };
}
function deleteDataByKey(value){
        var transaction=db.transaction(storeName,'readwrite');
        var store=transaction.objectStore(storeName);
        store.delete(value);
    }
function clearObjectStore(){
        var transaction=db.transaction(storeName,'readwrite');
        var store=transaction.objectStore(storeName);
        store.clear();
}
```

希望通过教辅帮助大家更好的理解与运行。