

HTML5 indexedDB前端本地存储数据库实例教程

这篇文章发布于 2017年07月20日, 星期四, 02:19, 归类于 [JS实例](#)。阅读 28146 次, 今日 60 次 16 条评论

by zhangxinxu from <http://www.zhangxinxu.com/wordpress/?p=6289>

本文可全文转载, 但需得到原作者书面许可, 同时保留原作者和出处, 摘要引流则随意。

// zxx: 本文内容较多, 有一定的深度, 建议预留足够的时间阅读, 或者可以先马后看~

一、indexedDB为何替代了Web SQL Database?

跟小朋友的教育从来没有有什么“赢在起跑线”这种说法一样, 在前端领域, 也不是哪来先出来哪个就在日后引领风骚的。

HTML5 indexedDB和Web SQL Database都是本地数据库数据存储, Web SQL Database数据库要出来的更早, 然并卵。从2010年11月18日W3C宣布舍弃Web SQL database草案开始, 就已经注定Web SQL Database数据库是明日黄花。

未来一定是indexedDB的, 从目前浏览器的兼容性来看, 也表明了这种结果:

IndexedDB - REC												
IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Chrome for Android	Browser for Android	QQ Browser	
		52	49			9.3		4.4				
	14	53	58		45	10.2		4.4.4				
11	15	54	59	10.1	46	10.3	all	56	59	11.4	1.2	
	16	55	60	11	47	11						
		56	61	TP	48							
		57	62									

张 <http://www.zhangxinxu.com>
鑫旭-鑫空间-鑫生活

Web SQL Database - UNOFF												
IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Chrome for Android	Browser for Android	QQ Browser	
		52	49			9.3		4.4				
	14	53	58		45	10.2		4.4.4				
11	15	54	59	10.1	46	10.3	all	56	59	11.4	1.2	
	16	55	60	11	47	11						
		56	61	TP	48							
		57	62									

可以看到IE和Firefox并不支持Web SQL Database, 基本上可以断定永远也不会支持, 规范都不认可, 实在没有浪费精力去支持的理由。

好, 现在我们知道了indexedDB取代Web SQL Database大局已定, 那大家知道为何Web SQL Database会被

舍弃吗?

下面是一段Web SQL Database代码:

```
database.transaction(function(tx) {  
    tx.executeSql("CREATE TABLE IF NOT EXISTS tasks (id REAL UNIQUE, text TEXT)", []);  
})
```

可以看到直接把SQL语句弄到JS中了, 主流的关系型数据库即视感, 这么设计看上去似乎无可厚非, 但恰恰这个设计成为了Web SQL Database被舍弃的重要原因之一: 一是学习成本高了很多, SQL虽然本身并不复杂, 但跨度较大, 例如我司玩JS的工程师和玩SQL的工程师中间还隔了个玩php的工程师; 二是本身使用很不方便, 需要把JS对象转换成关系型的字符串语句, 很啰嗦的。

而indexedDB直接JS对象入库, 无缝对接。

下表为更详细的indexedDB和Web SQL Database对比内容:

WebSQL
IndexedDB
优点
真正意义上的关系型数据库, 类似SQLite (SQLite是遵守ACID的轻型的数据库管理系统)
<ul style="list-style-type: none">允许对象的快速索引和搜索, 因此在Web应用程序场景中, 您可以非常快速地管理数据以及读取/写入数据。由于是NoSQL数据库, 因此我们可以根据实际需求设定我们的JavaScript对象和索引。在异步模式下工作, 每个事务具有适度的粒状锁。这允许您在JavaScript的事件驱动模块内工作。
不足
<ul style="list-style-type: none">规范不支持啦由于使用SQL语言, 因此我们需要掌握和转换我们的JavaScript对象为对应的查询语句。非对象驱动。
如果你的世界观里面只有关系型数据库, 恐怕不太容易理解。
位置
包含行和列的表。
包含JavaScript对象和键的存储对象。
查询机制
SQL
Cursor APIs, Key Range APIs, 应用程序代码
事务
锁可以发生在数据库, 表, 行的“读写”时候。
锁可以发生在数据库版本变更事务, 或是存储对象“只读”和“读写”事务时候。
事务提交
事务创建是显式的。默认是回滚, 除非我们调用提交。
事务创建是显式的。默认是提交, 除非我们调用中止或有一个错误没有被捕获。

STOP!

不能再继续说下去了, 估计看完上面对比表中的内容, 已经有很多同学已经有些不知所以然了, 因为其中涉及到了很多数据库相关的概念, 要想轻松理解上面内容以及更轻松掌握indexedDB的相关知识, 这些概念还是需要掌握的。

二、务必先了解数据库的一些概念

对于前端同学, 数据库概念没必要理解十分精准, 只需要知道大概怎么回事就好了, 因此, 下面的一些解释会其意, 勿嚼字。

1. 关系型数据库和非关系型数据库

“关系型数据库”是历史悠久，已经有半个世纪，占据主流江山的数据库模型，估计诸位公司的网站多半都是使用的这种数据库模型。而“非关系型数据库”（NoSQL数据库）则要年轻很多，根据资料显示是Carlo Strozzi在1998年提出来的，20年不到，使用键值对存储数据，且结构不固定，非常类似JavaScript中的纯对象。

```
var obj = {  
  key1: 'value1',  
  key2: 'value2',  
  key3: 'value3',  
  ...  
};
```

“关系型数据库”对一致性要求非常严格，例如要写入100个数据，前99个成功了，结果第100个不合法，此时事务会回滚到最初状态。这样保证事务结束和开始时候的存储数据处于一致状态。非常适合银行这种对数据一致性要求非常高的场景。但是，这种一致性保证是牺牲了一部分性能的。

但是，对于微博，QQ空间这类web2.0应用，一致性却不是显得那么重要，比方说朋友A看我的主页和朋友B看我的主页信息有不一样，没什么大不了的，有个几秒差异很OK的。虽然这类应用对一致性要求不高，但对性能要求却很高，因为读写实在太频繁了。如果使用“关系型数据库”，一致性的好处没怎么收益，反而性能问题比较明显，此时，不保证一致性但性能更优的“非关系型数据库”则更合适。同时，由于“非关系型数据库”的数据结构不固定，非常容易扩展。由于对于社交网站，需求变动那是一日三餐常有的事，添加新字段在所难免，“非关系型数据库”轻松上阵，如果使用“关系型数据库”，多半要数据大变动，要好好琢磨琢磨了。

从气质上讲，“关系型数据库”稳重持久，“非关系型数据库”迅速灵动。

在前端领域，Web SQL Database是“关系型数据库”，indexedDB是“非关系型数据库”。

2. 了解数据库中的事务-transaction

数据库的事务（英文为'transaction'），我们可以理解为对数据库的操作，而且专指一个序列上的操作。举个例子，银行转账，一个账号钱少了然后另一个账号钱多了，这两个操作要么都执行，要么都不执行。像这种操作就可以看成一个事务。

事务的提出主要是为了保证并发情况下保持数据一致性。关系型数据库中的事务具有下面4个基本特征：

1. 原子性(Atomicity)：事务中的所有操作作为一个整体提交或回滚。
2. 一致性(Consistency)：事物完成时，数据必须是一致的，以保证数据的无损。
3. 隔离性(Isolation)：对数据进行修改的多个事务是彼此隔离的。
4. 持久性(Durability)：事务完成之后，它对于系统的影响是永久的，该修改即使出现系统故障也将一直保留。

通常专业描述中会抽取这4个基本特性的首字母，统称为“**ACID特性**”。事务执行过程可以粗浅地理解为：开始事务，巴拉巴拉操作，如果错误，回滚（rollback），如果没问题，提交（commit），结束事务。

3. 了解数据库中的游标-cursor

现实世界中的“游标”相关联的常见事物就是“游标卡尺”，有刻度有区域：



数据库中的游标其实与之有共同之处。内存条本质上就像一把尺子，我们可以想象上面有很多刻度，然后内存大小就是由这些刻度一个一个堆砌起来的。数据库的事务为了保证数据可以回滚，显然需要有一片内存区域放置那些即将受影响是数据，这个内存区域中的虚表就是数据库的“游标”。

和现实世界的“游标卡尺”相映射就是：一个刻度表示一行数据，游标就是尺子上的一片区域，想要获得数据库一行一行的数据，我们可以遍历这个游标就好了。

4. 数据库中的“锁”-lock

数据库中的“锁”是保证数据库数据高并发时候数据一致性的一种机制。举个例子：现有两处火车票售票点，同时读取某发现上海到北京车票余额为 **5**。此时两处售票点同时卖出一张车票，同时修改余额为 **5-1** 也就是 **4** 写回数据库，这样就造成了实际卖出两张火车票而数据库中的记录却只少了 **1** 张。为了避免发生这种状况，就有了锁机制，也就是执行多线程时用于强行限制资源访问。

三、从简单实用的实例开始学习indexedDB

要想系统学习indexedDB相关知识，可以去[MDN文档](#)啃API，假以时日就可以成为前端indexedDB方面的专家。但是这种学习方法周期长，过于痛苦，无法立竿见影，因为API实在太多，天天花个把小时，估计也需要数周时间才能全部通透。

一想到投入产出比这么低，于是我决定从实例入手开始学习，能够实现基本的数据库增删改查功能就好，基本上80%+的相关需求都能从容搞定；等日后有机会再慢慢深入。

花了几个晚上折腾，一个具有增删改无查的实例页面终于弄出来了，您可以狠狠的点击这里：[HTML5 indexedDB存储编辑和删除数据demo](#)

我们第一次进去是没有数据的，显示如下：

1. 此时，我们填写表单，创建一条数据，如下：

点击“确定创建”按钮后，就得到如下图所示的结果：

此时我们刷新页面，依然是上图所示的结果。

打开控制台，在Application→indexedDB中，我们可以看到当前存储的数据字段和值等信息：

2. 我们可以直接对创建的数据进行修改，例如修改备注信息：

失焦后直接就修改了本地数据库信息了，我们刷新一下，可以看到备注文案已经变成“是个帅哥~”

3. 我们还可以对数据库数据进行删除，例如点击删除按钮：

结果这条数据库数据就被删掉了，界面重新显示为“暂无数据”。

以上就是demo页面所实现的indexedDB的增加数据，编辑数据和删除数据功能。

如果您当前的需求急用，可以把demo页面上的JS源代码直接复制过去改改；如果你希望对indexedDB相

关API及其使用有所了解，就继续往下看。

1. 首先打开indexedDB数据库

indexedDB数据库打开非常简单，语法就是字面意思：

```
window.indexedDB.open(dbName, version);
```

`dbName` 就是数据库名称，例如demo中使用的是 `'project'`；`version` 表示数据库的版本，根据我的理解，当我们对数据库的字段进行增加或修改时候，需要增加个版本。

通常，打开indexedDB数据库是和一些回调方法一起出现的，代码套路比较固定，基本上如果大家在实际项目中使用的话，都可以使用类似的代码：

```
// 数据库数据结果
var db;
// 打开数据库
var DBOpenRequest = window.indexedDB.open('project', 1);
// 数据库打开成功后
DBOpenRequest.onsuccess = function(event) {
    // 存储数据结果
    db = DBOpenRequest.result;
    // 做其他事情...
};

// 下面事情执行于：数据库首次创建版本，或者window.indexedDB.open传递的新版本（版本数值要比现在的高）
DBOpenRequest.onupgradeneeded = function(event) {
    // 通常对主键，字段等进行重定义，具体参见demo
};
```

其中，最重要就是红色高亮的 `db` 变量，我们后面所有的数据库操作都离不开它。

2. 创建indexedDB数据库的主键和字段

在我们开始数据库的增加删除操作之前，首先要把我们数据库的主键和一些字段先建好。是在 `onupgradeneeded` 这个回调方法中设置的，这个回调方法执行于数据库首次创建版本或者数据库 `indexedDB.open()` 方法中传递的版本号比当前版本号要高。

我们对数据库某一行数据进行增加删除操作，我们是没有必要对数据库的版本号进行修改的。但是对于字段修改就不一样了，比方说原来是 `5` 列数据，我们现在改成 `6` 列，由于相关设置是在 `onupgradeneeded` 回调中，因此，我们需要增加版本号来触发字段修改。demo页面这部分代码如下：

```
DBOpenRequest.onupgradeneeded = function(event) {
    var db = event.target.result;

    // 创建一个数据库存储对象
    var objectStore = db.createObjectStore(dbName, {
        keyPath: 'id',
        autoIncrement: true
    });

    // 定义存储对象的数据项
    objectStore.createIndex('id', 'id', {
        unique: true
    });
    objectStore.createIndex('name', 'name');
    objectStore.createIndex('begin', 'begin');
```

```
objectStore.createIndex('end', 'end');
objectStore.createIndex('person', 'person');
objectStore.createIndex('remark', 'remark');
};
```

这里出现了一个比较重要的概念，叫做 `objectStore`，这是indexedDB可以替代Web SQL Database的重要优势所在，我称之为“存储对象”。

`objectStore.add()` 可以向数据库添加数据，`objectStore.delete()` 可以删除数据，`objectStore.clear()` 可以清空数据库，`objectStore.put()` 可以替换数据等还有很多很多操作API。

在这里，我们使用 `objectStore` 来创建数据库的主键和普通字段。

`db.createObjectStore` 在创建主键同时返回“存储对象”，本演示中使用自动递增的 `id` 字段作为关键路径。`createIndex()` 方法可以用来创建索引，可以理解为创建字段，语法为：

```
objectStore.createIndex(indexName, keyPath, objectParameters)
```

其中：

`indexName`

创建的索引名称，可以使用空名称作为索引。

`keyPath`

索引使用的关键路径，可以使用空的 `keyPath`，或者 `keyPath` 传为数组 `keyPath` 也是可以的。

`objectParameters`

可选参数。常用参数之一是 `unique`，表示该字段值是否唯一，不能重复。例如，本demo中 `id` 是不能重复的，于是有设置：

```
objectStore.createIndex('id', 'id', {
  unique: true
});
```

3. indexedDB数据库添加数据

字段建好了之后，我们就可以给indexedDB数据库添加数据了。

由于数据库的操作都是基于事务（transaction）来进行，于是，无论是添加编辑还是删除数据库，我们都要先建立一个事务（transaction），然后才能继续下面的操作。语法就是名称：

```
var transaction = db.transaction(dbName, "readwrite");
```

`dbName` 就是数据库的名称，于是我们的数据库添加数据操作变得很简单：

```
// 新建一个事务
var transaction = db.transaction('project', "readwrite");
// 打开存储对象
var objectStore = transaction.objectStore('project');
// 添加到数据对象中
objectStore.add(newItem);
```

使用一行语句表示就是：

```
db.transaction('project', "readwrite").objectStore('project').add(newItem);
```

这里的 `newItem` 直接就是一个原生的纯粹的JavaScript对象，在本demo中，`newItem` 数据类似下面这样：

```
{
  "name": "第一个项目",
  "begin": "2017-07-16",
  "end": "2057-07-16",
  "person": "张鑫旭",
  "remark": "测试测试"
}
```

可以发现，当我们使用indexedDB数据库添加数据的时候，根本就不用去额外学习SQL语句，原始的JavaScript对象直接无缝对接，，

4. indexedDB数据库的编辑

原理为，先根据 `id` 获得对应行的存储对象，方法为 `objectStore.get(id)`，然后在原存储对象上进行替换，再使用 `objectStore.put(record)` 进行数据库数据替换。

代码示意：

```
function edit(id, data) {
  // 新建事务
  var transaction = db.transaction('project', "readwrite");
  // 打开已经存储的数据对象
  var objectStore = transaction.objectStore(project);
  // 获取存储的对应键的存储对象
  var objectStoreRequest = objectStore.get(id);
  // 获取成功后替换当前数据
  objectStoreRequest.onsuccess = function(event) {
    // 当前数据
    var myRecord = objectStoreRequest.result;
    // 遍历替换
    for (var key in data) {
      if (typeof myRecord[key] !== 'undefined') {
        myRecord[key] = data[key];
      }
    }
    // 更新数据库存储数据
    objectStore.put(myRecord);
  };
}
```

其中，`id` 就是要替换的数据库行的 `id` 字段值，因为唯一的主键，可以保证准确和高性能；`data` 则需要替换的数据对象，例如从“测试测试”修改为“是个帅哥~”，则调用：

```
edit(1, {
  remark: '是个帅哥~'
});
```

就可以了，跟我们平常写JS代码就没有什么区别。

5. indexedDB数据库的删除

和添加操作正好相反，但代码结构却是类似的，一行表示法：


```
db.transaction('project', "readwrite").objectStore('project').delete(id);
```

`id` 表示需要删除行 `id` 字段对应的值。

5. indexedDB数据库的获取

indexedDB数据库的获取使用Cursor APIs和Key Range APIs。

也就是使用“游标API”和“范围API”。在本文的演示页面中，只使用了“游标API”，直接显示全部数据，对于“范围API”并没有使用，但这里也会简单介绍下。

“游标API”可以让我们一行一行读取数据库数据，代码示意：

```
var objectStore = db.transaction(dbName).objectStore(dbName);
objectStore.openCursor().onsuccess = function(event) {
    var cursor = event.target.result;
    if (cursor) {
        // cursor.value就是数据对象
        // 游标没有遍历完，继续
        cursor.continue();
    } else {
        // 如果全部遍历完毕...
    }
}
```

可以看到，我们使用存储对象的 `openCursor()` 打开游标，在 `onsuccess` 回调中就可以遍历我们的游标对象了。其中 `cursor.value` 就是完整的数据对象，纯JS对象，就像下面这样：

```
{
  "id": 1,
  "name": "第一个项目",
  "begin": "2017-07-16",
  "end": "2057-07-16",
  "person": "张鑫旭",
  "remark": "测试测试"
}
```

我们就可以按照需求随意显示我们的数据了。

“范围API”是和“游标API”一起使用的，看下面这个例子，只打开 `id` 从4~10之间的数据，则有：

```
// 确定打开的游标的主键范围
var keyRangeValue = IDBKeyRange.bound(4, 10);
// 打开对应范围的游标
var objectStore = db.transaction(dbName).objectStore(dbName);
objectStore.openCursor(keyRangeValue).onsuccess = function(event) {
    var cursor = event.target.result;
    // ...
}
```

其中，有 `bound()`，`only()`，`lowerBound()`和 `upperBound()` 这几个方法，意思就是方法名字面意思，“范围内”，“仅仅是”，“小于某值”和“大于某值”。

方法最后还支持两个布尔值参数，例如：

```
IDBKeyRange.bound(4, 10, true, true)
```


则表示范围 `3~9`，也就是为 `true` 的时候不能和范围边界相等。

四、indexedDB存储和localStorage存储对比

- indexedDB存储IE10+支持，localStorage存储IE8+支持，后者兼容性更好；
- indexedDB存储比较适合键值对较多的数据，我之前不少项目需要存储多个字段，使用的是localStorage存储，结果每次写入和写出都要字符串化和对象化，很麻烦，如果使用indexedDB会轻松很多，因为无需数据转换。
- indexedDB存储可以在workers中使用，localStorage貌似不可以。这就使得在进行PWA开发的时候，数据存储的技术选型落在了indexedDB存储上面。

总结下就是，如果是浏览器主窗体线程开发，同时存储数据结构简单，例如，就存个 `true/false`，显然 `localStorage` 上上选；如果数据结构比较复杂，同时对浏览器兼容性没什么要求，可以考虑使用indexedDB；如果是在Service Workers中开发应用，只能使用indexedDB数据存储。

五、结束语

indexedDB数据库的使用目前可以直接在http协议下使用，这个和 `cacheStorage` 缓存存储必须使用https协议不一样。所以就应用场景来讲，indexedDB数据库还是挺广的，考虑到IE10也支持，所以基本可以确定在实际项目中应用是绝对不成问题的。

例如，页面中一些不常变动的结构化数据，我们就可以使用indexedDB数据库存储在本地，有助于增强页面的交互性能。`cacheStorage` 缓存页面，indexedDB数据库缓存数据，两者一结合而就可以实现百分百的离线开发，听上去还是很厉害的。

内容较多，时间较赶，表述有错误在所难免，欢迎大力指正。

感谢阅读，欢迎交流！

参考文章：

- [Migrating your WebSQL DB to IndexedDB](#)
- [数据库事务](#)
- [关系数据库与非关系数据库](#)
- [数据库——游标](#)

《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

（本篇完） // 想要打赏？点击[这里](#)。有话要说？点击[这里](#)。



« [借助Service Worker和cacheStorage缓存及离线开发](#)

[HTML5 file API加canvas实现图片前端JS压缩并上传](#) »

猜你喜欢

- 借助Service Worker和cacheStorage缓存及离线开发
- 翻译: 清除各个浏览器中的数据研究
- 突破本地离线存储5M限制的JS库localforage简介
- 翻译-你必须知道的28个HTML5特征、窍门和技术
- HTML5 localStorage本地存储实际应用举例
- 遐想: 如果没有IE6和IE7浏览器...
- reflection.js-实现图片投影倒影效果js插件
- HTML CSS列表元素ul,ol,dl的研究与应用
- 让所有浏览器支持HTML5 video视频标签
- CSS3&HTML5各浏览器支持情况一览表
- 基于原生HTML的UI组件开发

分享到: 1

标签: CacheStorage, HTML5, indexedDB, Service Workers, web SQL, 数据库, 本地存储, 离线存储

发表评论（目前16条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. 瑞君说道:

2018年04月9日 07:19

学习完这个后还特地写了一个操作 indexedDB 的 npm package, <https://www.npmjs.com/package/indexeddb-crud>

[回复](#)



2. 挣扎的小男孩说道:

2017年11月24日 22:31

目前正在学习, 之前也一直在看你的文章, 文章写的很明白。希望转载。

[回复](#)



3. 猫咪君-VRLie说道:

2017年10月18日 10:19

今年年初的时候试着弄了一下, indexedDB, 当时被它的异步机制给坑坏了

[回复](#)



4.

恒娃说道:

2017年10月16日 11:09

第二章节 第4小结 , 上海到背景车票, 打错字了。

回复



5.

sima说道:

2017年09月19日 11:21

赞

回复




6.

轻键快码说道:

2017年08月23日 10:55

补充一下 localStorage 是同步, 阻塞的存储机制, IndexedDB 是异步的.

回复



7.

isLishude说道:

2017年08月2日 13:50

这个很赞! 锁和游标的解释言简意赅

回复



8.

木叶说道:

2017年07月27日 14:32

修改的数据不会被保存下来, 只有新建和删除有用。在谷歌 58.0.3029.110 版本上试的

回复

木叶说道:

2017年07月27日 15:00

应该是执行 method.edit(id, data); 时这里的 id 变量是字符串而不是number类型造成的

回复

mike说道:

2017年08月25日 11:05

我也遇到这个问题了,按你的解决方案问题解决了 谢谢.
但是原理是啥~~

回复




9.

前端小武说道:

2017年07月27日 12:59

m

回复




10.

ghost说道:

2017年07月25日 10:48

马克

回复



11. **r**说道:
2017年07月20日 16:05
看来最近在研究 PWA
[回复](#)



12. **ClayIdols**说道:
2017年07月20日 09:13
我一直都是用 Dexie.js 来操作 indexedDB 的, 很方便
[回复](#)



- r**说道:
2017年07月20日 16:07
localforage 非常方便, 还支持优雅降级, Promise 操作 建议试试
[回复](#)



- ClayIdols**说道:
2017年07月20日 19:10
嗯, 看了简介不错哈, 实际用用看看效果怎么样
[回复](#)



最新文章

- » [常见的CSS图形绘制合集](#)
- » [粉丝群第1期CSS小测点评与答疑](#)
- » [分享三个纯CSS实现26个英文字母的案例](#)
- » [小tips: 纯CSS实现打字动画效果](#)
- » [CSS/CSS3 box-decoration-break属性简介](#)
- » [CSS :placeholder-shown伪类实现Material Design占位符交互效果](#)
- » [从天猫某活动视频不必要的3次请求说起](#)
- » [CSS vector-effect与SVG stroke描边缩放](#)
- » [CSS ::backdrop伪元素是干嘛用的?](#)
- » [周知: CSS -webkit-伪元素选择器不再导致整行无效](#)

今日热门

- » [常见的CSS图形绘制合集](#) ⁽¹⁹⁰⁾
- » [未来必热: SVG Sprite技术介绍](#) ⁽¹¹⁹⁾
- » [粉丝群第1期CSS小测点评与答疑](#) ⁽¹¹⁵⁾
- » [HTML5终极备忘大全 \(图片版+文字版\)](#) ⁽⁹³⁾
- » [让所有浏览器支持HTML5 video视频标签](#) ⁽⁸⁶⁾
- » [Selectivizr-让IE6~8支持CSS3伪类和属性选择器](#) ⁽⁸²⁾
- » [CSS3下的147个颜色名称及对应颜色值](#) ⁽⁷⁹⁾
- » [视区相关单位vw, vh..简介以及可实际应用场景](#) ⁽⁷⁶⁾
- » [写给自己看的display: flex布局教程](#) ⁽⁷⁶⁾
- » [小tips: 纯CSS实现打字动画效果](#) ⁽⁷⁶⁾

今年热议

- » 《CSS世界》女主角诚寻靠谱一起奋斗之人 ⁽⁷⁶⁾
- » 不借助Echarts等图形框架原生JS快速实现折线图效果 ⁽⁶⁴⁾
- » 看，for..in和for..of在那里吵架！ ⁽⁶⁰⁾
- » 是时候好好安利下LuLu UI框架了！ ⁽⁴⁷⁾
- » 原来浏览器原生支持JS Base64编码解码 ⁽³⁵⁾
- » 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 ⁽³³⁾
- » 炫酷H5中序列图片视频化播放的高性能实现 ⁽³¹⁾
- » CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 ⁽³⁰⁾
- » windows系统下批量删除OS X系统.DS_Store文件 ⁽²⁶⁾
- » 写给自己看的display: flex布局教程 ⁽²⁶⁾

猜你喜欢

- 借助Service Worker和localStorage缓存及离线开发
- 翻译：清除各个浏览器中的数据研究
- 突破本地离线存储5M限制的JS库localforage简介
- 翻译-你必须知道的28个HTML5特征、窍门和技术
- HTML5 localStorage本地存储实际应用举例
- 遐想：如果没有IE6和IE7浏览器...
- reflection.js-实现图片投影倒影效果js插件
- HTML CSS列表元素ul,ol,dl的研究与应用
- 让所有浏览器支持HTML5 video视频标签
- CSS3&HTML5各浏览器支持情况一览表
- 基于原生HTML的UI组件开发