你知道的浏览器存储有哪些，他们的区别是什么？

随着移动网络的发展与演化，我们手机上现在除了有原生 App，还能跑“WebApp”——它即开即用，用完即走。一个优秀的 WebApp 甚至可以拥有和原生 App 媲美的功能和体验。WebApp 优异的性能表现，有一部分原因要归功于浏览器存储技术的提升。cookie 存储数据的功能已经很难满足开发所需，逐渐被 WebStorage、IndexedDB 所取代，本文将介绍这几种存储方式的差异和优缺点。

HTML5 中新引入的浏览器存储方式 localStorage，sessionStorage，indexDB 接下来我们挨个看一看。

LocalStorage

1. LocalStorage 的特点

保存的数据长期存在，下一次访问该网站的时候，网页可以直接读取以前保存的数据。
大小为 5M 左右
仅在客户端使用，不和服务端进行通信
接口封装较好

基于上面的特点，LocalStorage 可以作为浏览器本地缓存方案，用来提升网页首屏渲染速度(根据第一请求返回时，将一些不变信息直接存储在本地)。

2. 存入/读取数据

localStorage 保存的数据，以“键值对”的形式存在。也就是说，每一项数据都有一个键名和对应的值。所有的数据都是以文本格式保存。
存入数据使用 setItem 方法。它接受两个参数，第一个是键名，第二个是保存的数据。

```
localStorage.setItem("key","value");
```

读取数据使用 getItem 方法。它只有一个参数，就是键名。

```
var valueLocal = localStorage.getItem("key");
```

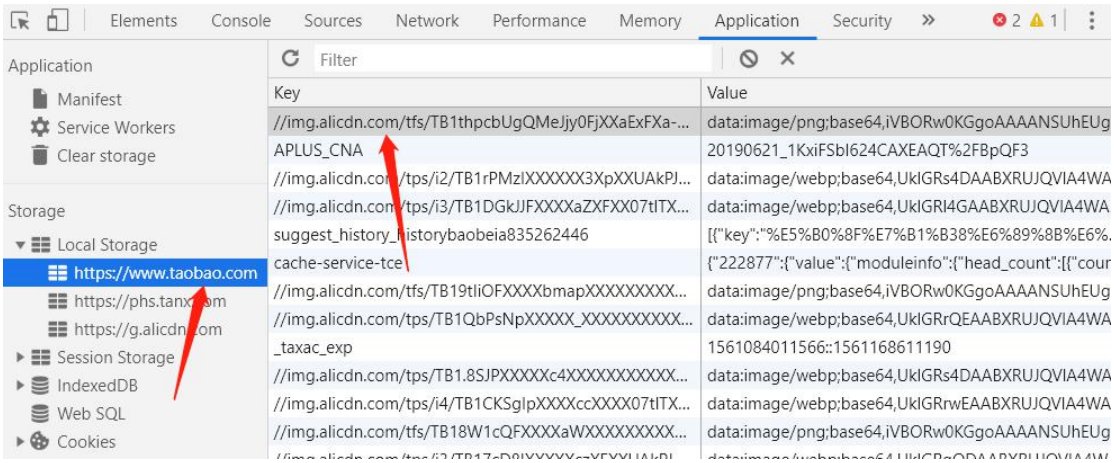
具体代码：

```
<body>
<div id="name"></div>
<div id="gender"></div>
<script>
var name = localStorage.getItem("name");
var gender = localStorage.getItem("gender");
document.getElementById("name").innerHTML = name;
```

```
document.getElementById("gender").innerHTML = gender;
</script>
</body>
```

3. 使用场景

LocalStorage 在存储方面没有什么特别的限制，理论上 Cookie 无法胜任的、可以用简单的键值对来存取的数据存储任务，都可以交给 LocalStorage 来做。这里给大家举个例子，考虑到 LocalStorage 的特点之一是持久，有时我们更倾向于用它来存储一些内容稳定的资源。比如图片内容丰富的电商网站会用它来存储 Base64 格式的图片字符串：



sessionStorage

sessionStorage 保存的数据用于浏览器的一次会话，当会话结束（通常是该窗口关闭），数据被清空；sessionStorage 特别的一点在于，**即便是相同域名下的两个页面，只要它们不在同一个浏览器窗口中打开，那么它们的 sessionStorage 内容便无法共享**；localStorage 在所有同源窗口中都是共享的；cookie 也是在所有同源窗口中都是共享的。除了保存期限的长短不同，SessionStorage 的属性和方法与 LocalStorage 完全一样。

1. sessionStorage 的特点

会话级别的浏览器存储
大小为 5M 左右
仅在客户端使用，不和服务端进行通信
接口封装较好

基于上面的特点，sessionStorage 可以有效对表单信息进行维护，比如刷新时，表单信息不丢失。

2. 使用场景

sessionStorage 更适合用来存储生命周期和它同步的会话级别的信息。这些信息只适用于当前会话，当你开启新的会话时，它也需要相应的更新或释放。

3.sessionStorage 、localStorage 之间的区别

共同点：都是保存在浏览器端，且都遵循同源策略。
不同点：在于生命周期与作用域的不同

作用域：localStorage 只要在相同的协议、相同的主机名、相同的端口下，就能读取/修改到同一份 localStorage 数据。sessionStorage 比 localStorage 更严苛一点，除了协议、主机名、端口外，还要求在同一窗口（也就是浏览器的标签页）下

生命周期：localStorage 是持久化的本地存储，存储在其中的数据是永远不会过期的，使其消失的唯一办法是手动删除；而 sessionStorage 是临时性的本地存储，它是会话级别的存储，当会话结束（页面被关闭）时，存储内容也随之被释放。

Web Storage 是一个从定义到使用都非常简单的东西。它使用键值对的形式进行存储，这种模式有点类似于对象，却甚至连对象都不是——它只能存储字符串，要想得到对象，我们还需要先对字符串进行一轮解析。

Web Storage 它只能用于存储少量的简单数据。当遇到大规模的、结构复杂的数据时，Web Storage 也爱莫能助了。这时候我们就要清楚我们的终极大 boss——IndexedDB！

IndexedDB

IndexedDB 是一种低级 API，**用于客户端存储大量结构化数据(包括文件和 blobs)**。该 API 使用索引来实现对该数据的高性能搜索。IndexedDB 是一个运行在浏览器上的非关系型数据库。既然是数据库了，那就不是 5M、10M 这样小打小闹级别了。理论上来说，IndexedDB 是没有存储上限的（一般来说不会小于 250M）。它不仅可以存储字符串，还可以存储二进制数据。

1. IndexedDB 的特点

键值对储存。

IndexedDB 内部采用对象仓库（object store）存放数据。所有类型的数据都可以直接存入，包括 JavaScript 对象。对象仓库中，数据以"键值对"的形式保存，每一个数据记录都有对应的主键，主键是独一无二的，不能有重复，否则会抛出一个错误。

- 异步

IndexedDB 操作时不会锁死浏览器，用户依然可以进行其他操作，这与 LocalStorage 形成对比，后者的操作是同步的。异步设计是为了防止大量数据的读写，拖慢网页的表现。

- 支持事务。

IndexedDB 支持事务（transaction），这意味着一系列操作步骤之中，只要有一步失败，整个事务就都取消，数据库回滚到事务发生之前的状态，不存在只改写一部分数据的情况。

- 同源限制

IndexedDB 受到同源限制，每一个数据库对应创建它的域名。网页只能访问自身域名下的数据库，而不能访问跨域的数据库。

- 储存空间大

IndexedDB 的储存空间比 LocalStorage 大得多，一般来说不少于 250MB，甚至没有上限。

- 支持二进制储存。

IndexedDB 不仅可以储存字符串，还可以储存二进制数据（ArrayBuffer 对象和 Blob 对象）。

2. IndexedDB 的常见操作

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。

- 建立打开 IndexedDB ----window.indexedDB.open("testDB")

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

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

```
<script>
function openDB(name) {
var request = window.indexedDB.open(name); //建立打开 IndexedDB
request.onerror = function (e) {
console.log("open indexdb error");
};
request.onsuccess = function (e) {
myDB.db = e.target.result; //这是一个 IDBDatabase 对象，这就是
IndexedDB 对象
console.log(myDB.db); //此处就可以获取到 db 实例
};
}
var myDB = {
name: "testDB",
version: "1",
db: null
};
openDB(myDB.name);
</script>
```

控制台得到一个 IDBDatabase 对象，这就是 IndexedDB 对象

- 关闭 IndexedDB----indexdb.close()

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

- 删除 IndexedDB----window.indexedDB.deleteDatabase(indexdb)

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

。

总结

正是浏览器存储、缓存技术的出现和发展,为我们的前端应用带来了无限的转机。近年来基于存储、缓存技术的第三方库层出不绝，此外还衍生出了 PWA 这样优秀的 Web 应用模型。总结下本文几个核心观点：

- Web Storage 是 HTML5 专门为浏览器存储而提供的数据存储机制，不与服务端发生通信
- IndexedDB 用于客户端存储大量结构化数据