sessionStorage 、localStorage 和 cookie 之间的共同点：

都是保存在浏览器端，且同源的。

sessionStorage 、localStorage 和 cookie 之间的区别：

cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递。而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下。

存储大小限制也不同，cookie数据不能超过4k，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。

作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；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。

Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。

Web Storage 的 api 接口使用更方便。

LocalStorage和sessionStorage都支持setItem。removeItem，clear。改变值时都响应storage事件。在ie9及以上支持storage事件，IE的事件绑定在document上document.attachEvent(‘onstorage’, callback)，其他浏览器绑定在window上window.addEventListener("storage", callback, false);

IE中的storageEvent对象不包含key/newValue/oldValue等属性，因此如果想知道是哪个Key的数据发生了变化需要自己处理，其他浏览器则可以直接获得数据。

IE9下在事件触发时尽然无法立即获取到对应key的值，需要使用setTimeout做异步处理。其他浏览器状况良好。

IE8/IE9/Firefox3.6: 在页面A中注册onstorage事件，修改localStorage时，A页面和其他页面都能收到onstorage事件。因此，对于这些浏览器监听onstorage时需要自己判断是否是本页面触发的，并且忽略本页面触发的行为。

Chrome12/Firefox4/Opera11/Safari5中只有收到由其他页面触发的onstorage事件。

此外，Chrome14 DEV版本中测试发现，在页面设置了document.domain之后，onstorage事件无论如何都不会触发，此Bug导致在Chrome下无法使用onstorage事件

storage回调函数中的返回参数属性

* storageArea: 表示存储类型（Session或Local）
* key:发生改变项的key
* oldValue: key的原值
* newValue: key的新值
* url\*: key改变发生的URL

\* 注意: url 属性早期的规范中为uri属性。有些浏览器发布较早，没有包含这一变更。为兼容性考虑，使用url属性前，你应该先检查它是否存在，如果没有url属性，则应该使用uri属性

如果调用clear()方法，那么key、oldValue和newValue都会被设置为null。