## 存储相关

### cookies sessionStorage和localstorage区别

相同点：都存储在客户端 不同点：1.存储大小

· cookie数据大小不能超过4k。

· 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。

2.有效时间

· localStorage    存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；

· sessionStorage  数据在当前浏览器窗口关闭后自动删除。

· cookie          设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

1. 数据与服务器之间的交互方式

· cookie的数据会自动的传递到服务器，服务器端也可以写cookie到客户端

· sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

### localStorage如何设置过期时间 如何封装自定义localStorage

1. 创建Storage类

* 定义 对应的get set remove clear api
* 通过set函数添加过期时间参数来实现过期时间的记录
* 设置存储时存储当前值和过期时间
* get取值的时候先验证当前值是否存在 以及时间是否大于过期时间 如果存在且不大于过期时间既可返回对应的值否则返回空

class Storage {  
 constructor() {  
 this.storageName = 'expiredStorage'  
 }  
   
   
 /\*\*  
 \* 设置缓存  
 \* @param {string} name 缓存名称  
 \* @param {any} value 缓存的值  
 \* @param {any} expires 缓存过期时间(秒)   
 \*\*/  
 set(name, value, expires) {  
 const storages ={}  
 storages[name] = {  
 value,  
 expires: storages[name]  
 ? storages[name].expires  
 : expires === undefined  
 ? +new Date() + 31536000000 //默认365天 这个值可以自己修改  
 : expires \* 1000 + +new Date(),  
 };  
 localStorage.setItem(this.storageName, JSON.stringify(storages))  
 }  
   
 /\*\*  
 \* 获取缓存  
 \* @param {string} name 缓存名称  
 \*\*/  
 get(name) {  
 const storages = JSON.parse(localStorage.getItem(this.storageName))  
 try {  
 if (!storages[name]) {  
 // 不存在  
 return null;  
 }  
 console.log('log=====', storages[name].expires - new Date());  
 if (+new Date() > storages[name].expires) {  
 // 存在但过期  
 this.remove(name);  
 return null;  
 }  
 return storages[name].value;  
 } catch (error) {  
 console.log('[ControlStorage] the error message: get field failed\n', error);  
 }  
 }  
   
 /\*\*  
 \* 移除对应缓存  
 \* @param {string} name 缓存名称  
 \*\*/  
 remove(name) {  
 const storages = JSON.parse(localStorage.getItem(this.storageName))  
 try {  
 delete storages[name];  
 if (JSON.stringify(storages) === '{}') {  
 // 缓存字段为空对象时，删除该字段  
 this.clear();  
 return;  
 }  
 this.baseStorage.setItem(storages);  
 } catch (error) {  
 console.log('[ControlStorage] the error message: remove field failed\n', error);  
 }  
 }  
 /\*\*  
 \* 清除所有带过期时间的缓存  
 \*\*/  
 clear() {  
 localStorage.removeItem(this.storageName)  
 }  
}  
  
export default new Storage();

1. 实际调用 引入对应的Storage类 调用里面的方法传递对应参数

import storage from './js/storage.js'  
  
...  
setStorage() {  
 // 5秒过期  
 storage.set('name', 'ghkmmm', 5)  
},  
getStorage() {  
 console.log(storage.get('name'))  
},  
removeStorage() {  
 storage.remove('name')  
},