localStorage

1.localStorage的优势和局限

localStorage的优势

1）localStorage拓展了cookie的4K限制

2）localStorage会可以将第一次请求的数据直接存储到本地，这个相当于一个5M大小的针对于前端页面的数据库，相比于cookie可以节约带宽，但是这个却是只有在高版本的浏览器中才支持的

localStorage的局限

1）浏览器的大小不统一，并且在IE8以上的IE版本才支持localStorage这个属性

2）目前所有的浏览器中都会把localStorage的值类型限定为**string**类型，这个在对我们日常比较常见的JSON对象类型需要一些转换

3）localStorage在浏览器的隐私模式下面是不可读取的

4）localStorage本质上是对字符串的读取，如果存储内容多的话会消耗内存空间，会导致页面变卡

5）localStorage不能被爬虫抓取到

2.localStorage与sessionStorage的唯一一点区别就是localStorage属于永久性存储，而sessionStorage属于当会话结束的时候，sessionStorage中的键值对会被清空

3.localStorage的使用

1）首先在使用localStorage的时候，我们需要判断浏览器是否支持localStorage这个属性

if(！window.localStorage){

//如果想兼容ie7以下，就在这里使用userData

alert("浏览器不支持localstorage");

return false;

}else{

//主逻辑业务

}

2）localStorage写入的三种方法

if(！window.localStorage){

alert("浏览器不支持localstorage");

return false;

}else{

var storage=window.localStorage;

//写入a字段

storage["a"]=1;

//写入b字段

storage.b=1;

//写入c字段（常用）

storage.setItem("c",3);

console.log(typeof storage["a"]);

console.log(typeof storage["b"]);

console.log(typeof storage["c"]);

}

存储进去的是int类型，但是打印出来却是string类型，这个与localStorage本身的特点有关，**localStorage只支持string类型的存储**

4）localStorage的使用也是遵循同源策略的，所以不同的网站直接是不能共用相同的localStorage

5）localStorage读取的三种方法

if(!window.localStorage){

alert("浏览器支持localstorage");

}else{

var storage=window.localStorage;

//写入a字段

storage["a"]=1;

//写入b字段

storage.a=1;

//写入c字段

storage.setItem("c",3);

console.log(typeof storage["a"]);

console.log(typeof storage["b"]);

console.log(typeof storage["c"]);

//第一种方法读取

var a=storage.a;

console.log(a);

//第二种方法读取

var b=storage["b"];

console.log(b);

//第三种方法读取（常用）

var c=storage.getItem("c");

console.log(c);

}

官方推荐使用setItem和getItem

6）localStorage的更改

if(!window.localStorage){

alert("浏览器支持localstorage");

}else{

var storage=window.localStorage;

//写入a字段

storage["a"]=1;

//写入b字段

storage.b=1;

//写入c字段

storage.setItem("c",3);

console.log(storage.a);

// console.log(typeof storage["a"]);

// console.log(typeof storage["b"]);

// console.log(typeof storage["c"]);

/\*分割线\*/

**storage.a=4**;

console.log(storage.a);

}

7）localStorage的删除

将localStorage的**所有内容**清除

var storage=window.localStorage;

storage.a=1;

storage.setItem("c",3);

console.log(storage);

storage.**clear()**;

console.log(storage);

将localStorage中的**某个键值对**删除

var storage=window.localStorage;

storage.a=1;

storage.setItem("c",3);

console.log(storage);

storage.**removeItem("a")**;

console.log(storage.a);

8）localStorage的键获取

var storage=window.localStorage;

storage.a=1;

storage.setItem("c",3);

for(var i=0;i<storage**.length**;i++){

var key=storage.**key(i)**;

console.log(key);

}

使用key()方法，向其中出入索引即可获取对应的键

**4.JSON处理**

**一般我们会将JSON（在JS中是个对象，其实JS中全都是对象）存入localStorage中**，但是在localStorage会自动将localStorage转换成为字符串形式。

这个时候我们可以使用JSON.stringify()这个方法，来将JSON转换成为JSON字符串

示例：

if(!window.localStorage){

alert("浏览器支持localstorage");

}else{

var storage=window.localStorage;

var data={

name:'xiecanyong',

sex:'man',

hobby:'program'

};

var d=JSON.stringify(data);

storage.setItem("data",d);

console.log(storage.data);

}

读取之后要将JSON字符串转换成为JSON对象，使用JSON.parse()方法

var storage=window.localStorage;

var data={

name:'xiecanyong',

sex:'man',

hobby:'program'

};

var d=JSON.stringify(data);

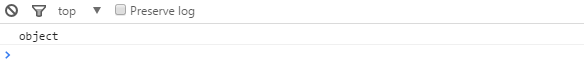
storage.setItem("data",d);

//将JSON字符串转换成为JSON对象输出

var json=storage.getItem("data");

var jsonObj=JSON.parse(json);

console.log(typeof jsonObj);



打印出来是Object对象

另外还有一点要注意的是，**其他类型读取出来也要进行转换**