知识点

> 链接：https://juejin.cn/post/6923331849708109838
>
> 这个感觉写的有点乱，不太清晰。

使用 `HTML5` 中的 `Web Storage API`， 可以在**客户端存储更多的数据**，，可以**实现数据在多个页面中共享甚至是同步**，对于复杂的数据，可以使用 Web SQL Database API 来实现。

掌握 `localStorage 和 sessionStorage` 存储方式， 存储 JSON 对象的数据， 使用 Web SQL Database 的基本操作。

Web Storage

`Web Storage API` 的属性，方法，事件。

`cookie` 可用于传递少量的数据， 是一个在服务器和客户端 之间来回传送文本值的内置机制，服务器可以根据 `cookie` 追踪 用户在不同页面的访问信息。

`cookie` 的特点：

第一，**大小的限制**，cookie 的大小 限制 在 4KB 以内。

第二，**带宽的限制**，cookie 数据 会在服务器和浏览器 之间来回传送，所以访问哪个页面，都会消耗网络的带宽。

第三，**安全风险**，因为 cookie 会频繁在网络中传送， 所以在网络上是可见的，在不加密的情况下，是由安全风险的。

第四，**操作复杂**，在客户端的浏览器中，使用 JavaScript 操作 cookie数据是比较复杂的。

所以，如果对于**较小的数据**，并且需要**在服务器和客户端之间**频繁传送时，才有 cookie 存在的意义。

什么是 web storage

`Web Storage` 可以在客户端保存大量的数据， `Web Storage` 本地存储的优势：

第一，存储容量大。

第二，零带宽。Web Storage 中的数据仅仅存储在本地，不会与服务器发生任何交互行为，不存在网络带宽的占用问题。

第三，编程接口。提供了一套丰富的接口，使得数据操作更加方便。

第四，独立的存储空间。每个域都有独立的存储空间，各个存储空间是完全能独立的，不会造成数据的混乱。

localStorage 和 sessionStorage

在 `Web Storage` 本地存储 包括 `sessionStorage` 会话存储 和 `localStorage` 本地存储。

`cookie 和 session` 完全是服务器端可以操作的数据， `sessionStorage 和 localStorage` 完全是浏览器端操作的数据。

`cookie 和 session` 完全继承同一个 `Storage API`，所以 `sessionStorage 和 localStorage` 的编程接口是一样的。

`sessionStorage 和 locatlStorage` 区别在于 数据存在**时间范围** 和 **页面范围。**

`sessionStorage`：数据只**保存到存储它的窗口或标签关闭时**，数据在构建它们的窗口或标签内也可见

`localStorage`：数据的生命周期比窗口或浏览器的生命周期长，数据可被同源的每个窗口或者标签共享。

监测是否支持 Web Storage

示例：

```
function CheckStorageSupport() {
 // 监测 sessionStorage
 if(window.sessionStorage) {
  console.log(“浏览器支持sessionStorage”);
 }else{
  console.log("浏览器不支持sessionStorage");
 }

 // 监测localStorage
 if(window.localStorage) {
  console.log("浏览器支持localStorage");
 }else {
  console.log("浏览器不支持localStorage");
 }
}
```

设置和获取Storage数据

保存数据到`sessionStorage`：

```
  window.sessionStorage.setItem("key”,"value");
```

`setItem()`表示保存数据的方法

从`sessionStorage`中获取数据：

```
  value = window.sessionStorage.getItem("key");
```

`getItem()`为获取数据的方法

保存数据的写法：

```
  window.sessionStorage.key = "value";
```

或

```
  window.sessionStorage["key"] = "value"
```

获取数据的方法更加直接：

```
value = window.sessionStorage.key;
```

或

```
value = window.sessionStorage["key"]
```

使用sessionStorage 和 localStorage

示例:

```
function DaDa() {
 window.localStorage.setItem("localKey", "localVlaue");

 // 获取
 console.log(window.localStorage.getItem("localKey"));

 window.sessionStorage.setItem("sessionKey", "sessionValue");

 // 获取
 console.log(window.sessionStorage.getItem("sessionKey"));
}
```

Storage接口

示例:

```
interface Storage{
 readonly attribute unsigned long length;
 DOMString ? key(unsigned long index);
 getter DOMString getItem(DOMString key);
 setter creator void setItem(DOMString key, DOMString value);
 deleter void removeItem(DOMString key);
 void clear();
}
```

1. `length`属性，表示当前`storage`对象中存储的键/值对的数量。

`Storage`对象是同源的，`length`属性只能反映同源的键/值对数量

1. `key`方法，获取指定位置的键。
2. `getItem`方法，根据键返回相应的数据值。
3. `setItem`方法，将数据存入指定键对应的位置。
4. `removeItem`方法，从存储对象中移除指定的键/值对。
5. `clear`方法，清除`Storage`对象中所有的数据，如`Storage`对象是空的，则不执行任何操作。

使用Storage对象保存页面的内容

示例:

```
// 保存数据到sessionStorage
```

```
function SaveStorage(frm) {
 var storage = window.sessionStorage;

 storage.setItem("name",frm.name.value);

 storage.setItem("age", frm.age.value);
}

遍历显示sessionStorage中的数据

function Show(){
 var storage = window.sessionStorage
 var result = "";
 for(var i=0; i<storage.length; i++){
  var key = storage.key(i);
  var value = storage.getItem(key);
  result += key + ":" + value + ";";
 }
}
```

存储JSON对象的数据

`Storage` 是以字符串保存数据的，所以在保存 `JSON` 格式的数据之前，需要把 `JSON` 格式的数据转化为字符串，这个操作叫**序列化。**

使用 `JSON.stringify()` 序列化 `json` 格式的数据为字符串数据:

```
var dada = JSON.stringify(jsonObject);
```

把数据反序列化为 `JSON` 格式：

```
var jsonObject = JSON.parse(stringData);
```

`web Storage` 建立一套会在数据更新时触发的事件通知机制，无论监听的窗口是否存储过该数据，只要与执行存储的窗口是同源的，都会触发 `web Storage` 事件。

```
window.addEventListener("storage", EventHandle, true);
```

`StorageEvent` 事件接口：

```
interface StorageEvent:Event {
 readonly attribute DOMString key;
 readonly attribute DOMString ? oldValue;
 readonly attribute DOMString ? newValue;
 readonly attribute DOMString ? url;
 readonly attribute Storage? storageArea;
}
```

`key` 属性：包含了存储总被更新或删除的键； `oldValue` 属性：包含了更新前键对应的数据。

`newValue`属性：包含了更新后的数据；`url`属性：指向`Storage`事件的发生源。

`storageArea`属性：该属性是一个引用，指向值发生改变的`localStorage`或`sessionStorage`。

web SQL Database

`Web SQL Database`使用的是 SQLite 数据库，允许应用程序通过一个异步的 JavaScript 接口访问数据库。SQLite是一款轻型的数据库。

1. `openDatabase()`方法，使用现有的数据库或新建数据库来创建数据库对象。
2. `transaction()`方法，允许设计者控制事务的提交或回滚。
3. `executeSql()`方法，用于执行真实的SQL查询。

操作Web sql数据库

```
var db = openDatabase("TestDB", "1.0", "测试", xxxx)
```

共5个参数：

1. 数据库名
2. 版本号
3. 数据库的描述
4. 数据库的大小
5. 创建回调函数

创建数据表

`transaction()`方法用于进行事务处理，`executeSql()`方法用于执行sql语句。

创建数据表：

```
db.transaction(function (tx) {
 tx.executeSql('CREATE TABLE IF NOT EXISTS UserName (id unique, Name)');
});
```

> `transaction()`方法传递给回调函数的tx是一个`transaction`对象，使用`transaction`对象的`executeSql()`方法可以执行SQL语句。

添加数据到数据表：

```
db.transaction(function (tx){
 tx.executeSql('INSERT INTO UserName (id,Name) VALUES(1,'dada'));
 tx.executeSql('INSERT INTO UserName (id,Name) VALUES (2,'dadada'));
});
```

读取数据库中的数据：

```
db.transaction(function (tx){
 tx.executeSql('SELECT * FROM UserName', [], function(tx, resultes){
  var len = results.rows.length;
  for (var i=0; i<len; i++){
   console.log(results.rows.item(i).Name);
  }
 },null);
});
```

html5几种存储形式

1. 本地存储-`localStorage, sessionStorage`
2. 离线缓存 `application cache`
3. `indexedDB 和 webSQL`

localStorage 和 sessionStorage

- `localStorage`永久存储，永不失效除非手动删除
- `sessionStorage`浏览器重新打开后就消失了

大小，每个域名是`5`M，cookie第一是由大小限制，大概4K左右，第二，ie6域名下有个数限制。

HTML API

在浏览器的API有两个，`localStorage和sessionStorage`

`window`对象中：`localStorage`对应`window.localStorage`，`sessionStorage`对应`window.sessionStorage`。

`localStorage`只要在相同的协议、相同的主机名、相同的端口下，就能读取或修改到同一份`localStorage`的数据。

`sessionStorage`比`localStorage`更严格，除了协议、主机名、端口外，**还要求在同一窗口下**

方法及含义：

```
setItem(‘key’,‘value’)  储存数据

getItem(‘key’)  读取数据

removeItem(‘key’)   删除数据

clear() 清空数据
```

使用方法详解：

```
//储存数据
window.localstage.setItem('key','value')
// key : 数据的名称
// value : 数据
// 所存储的是数据必须是string类型
```

```
//读取数据
window.localstage.getItem('key')
// key : 数据的名称
// 如果数据不存在，返回null （一般用它做判断）

//删除数据
window.localstage.removeItem('key')
// key : 数据的名称
// 删除本地存储中的指定数据

//清空数据
window.localstage.clear()
// 清空本地存储中的所有数据
```

什么是localStorage和sessionStorage?

> `localStorage`(长期存储)、`sessionStorage`(会话存储)是`H5`中的本地`web`存储提供的两个接口, 相当于前端一个小型的本地数据库，用于在本地存储一些不敏感的数据，隶属于`window`对象。

localStorage和sessionStorage的异同?

相同点:两者的`API`完全相同，都是在浏览器端存储数据。

不同点:

1. `localStorage`存储的数据是永久性数据,页面刷新，即使浏览器重启，甚至操作系统重启也不会丢失，并且存储的数据在同源（协议、域名、端口号一致）下的标签页和`window`窗口之间共享。
2. `sessionStorage`存储的数据有些苛刻，页面刷新仍然有效,选项卡关闭时数据丢失。但是在相同标签页打开的多个`iframe`之间数据可以共享(同源的情况下)。

> 两者都是在浏览器端存储数据,`localStorage`存储的数据被限制在同源下,可跨窗口通信,不可跨浏览器,跨域；`sessionStorage`存储的数据被限制在标签页(页卡关闭丢失)。

localStorage的局限

局限:

1.各个浏览器的支持的大小不一样，业界认为是`5M`,并且在不同的浏览器不同版本支持度不一样

2.`localStorage`的数据不能被爬虫抓取