# 六 拖拽

在HTML5的规范中，我们可以通过为元素增加draggable="true"来设置此元素是否可以进行拖拽操作，其中图片、链接默认是开启的。但是这时候的拖动没有带走数据，可以设置ondragstart事件就可以携带数据。

**属性**

a. draggable

**拖拽元素** 页面中设置了draggable="true"属性的元素

ondrag 应用于拖拽元素，整个拖拽过程都会调用

ondragstart 应用于拖拽元素，当拖拽开始时调用

ondragleave 应用于拖拽元素，当鼠标离开拖拽元素时调用

ondragend 应用于拖拽元素，当拖拽结束时调用

**目标元素** 页面中任何一个元素都可以成为目标元素

ondragenter 应用于目标元素，当拖拽元素进入时调用

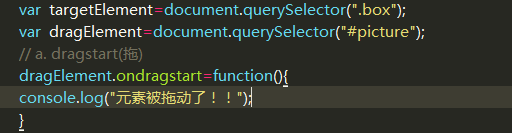
ondragover 应用于目标元素，当停留在目标元素上时调用

ondrop 应用于目标元素，当在目标元素上松开鼠标时调用

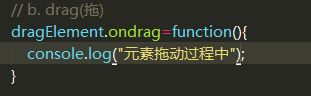
ondragleave 应用于目标元素，当鼠标离开目标元素时调用

**代码解释**

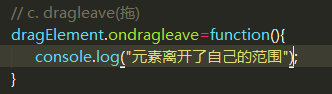
**a**. dragstart(拖) 只执行一次



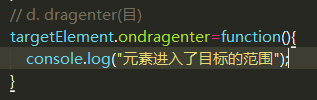
**b**. drag(拖) 只要拖动元素，一直执行



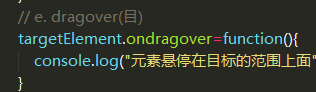
**c**. dragleave(拖) 只执行一次 当鼠标离开才会触发



**d.** dragenter(目) 只触发一次 进入一次触发一次

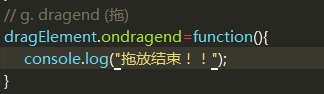


**e.** dragover(目) 只要停在上面一直执行



**f.** drop(目)

**g**. dragend (拖)



数据传递方法：

**a.** ev.dataTransfer.setData(key,value)

**b.** ev.dataTransfer.getData(key)

# 七 历史

提供window.history，对象我们可以管理历史记录，可用于单页面应用（Single Page Application），可以无刷新改变网页内容。

旧版本浏览器..

history.back() 回退

history.forward() 前进

history.length 历史条数

## 1 HTML5新增的历史方法

历史中添加一条数据：

pushState(data, title, url)

data 用于存储自定义数据，通常设为null

title 网页标题，基本上没有被支持，一般设为空

url 以当前域为基础增加一条历史记录，不可跨域设置

历史中替换一条数据：

replaceState() 与pushState()基本相同

不同：replaceState()只替换当前url，不增加/减少历史记录。

## 2 HTML5新增的历史事件

onpopstate当前进或后退时则触发，通过事件对象ev.state可以读取到存储的数据。

# 八 地理定位

在HTML规范中，增加了获取用户地理信息的API，这样使得我们可以基于用户位置开发互联网应用，即基于位置服务 (Location Base Service)

HTML5 Geolocation(地理位置定位) 规范提供了一套保护用户隐私的机制。必须先得到用户明确许可，才能获取用户的位置信息。

### API详解

navigator.geolocation.getCurrentPosition(successCallback,errorCallback, options) 获取当前地理信息

successCallback为方法成功的回调，此参数必须

errorCallback为方法失败时候的回调，此参数可选

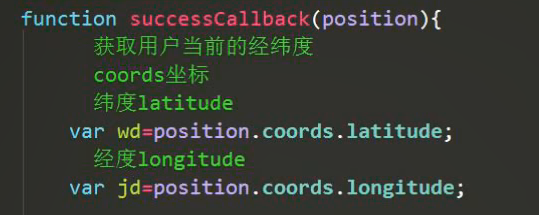
options参数为额外参数，也是可选参数，对象

navigator.geolocation.watchPosition(successCallback, errorCallback, options) 重复不停的获取当前最新的地理信息

1、当成功获取地理信息后，会调用succssCallback(position)，并返回一个包含位置信息的对象position。Coords(坐标)

position.coords.latitude纬度

position.coords.longitude经度



2、当获取地理信息失败后，会调用errorCallback，并返回错误信息error

3、可选参数 options 对象可以调整位置信息数据收集方式

# 九 Web存储

web存储即把数据存储在浏览器中。传统方式我们以document.cookie来进行存储的，但是由于其存储大小只有4k左右，并且解析也相当的复杂，HTML5规范则提出解决方案：Storage 存储（WebSQL、IndexDB

已经被w3c 放弃了）。分别有两种存储方式

会话存储：window.sessionStorage

本地存储：window.localStorage

注意：只能存储字符串，可以将对象JSON.stringify() 编码后存储，向本地保存数据,有可能在浏览器内存里面，有可能在硬盘上面。

## 1 会话存储

Web中的Session指的就是用户在浏览某个网站时，从进入网站到关闭浏览器所经过的这段时间，也就是用户浏览这个网站所花费的时间。

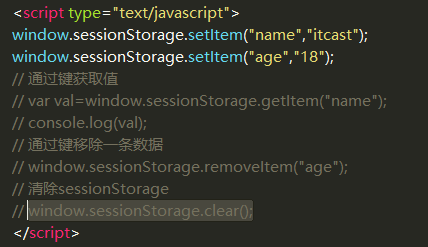
查看方式：Chrome开发者工具中Application

window.sessionStorage

1、生命周期为关闭浏览器窗口

2、在同一个窗口下数据可以共享

3、可存储大约5M



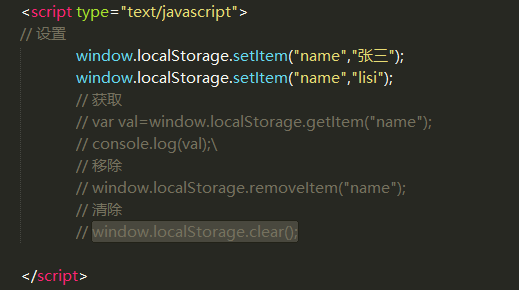
## 2 本地存储

window.localStorage

1、永久生效，除非手动删除

2、可以多窗口共享

3、可翻出大约20M



## 3 会话存储于本地存储区别

**sessionStorage生命周期从浏览器打开到结束、**

**localStorages生命周期永久保存在浏览器中，除非清除移除**

# 十 全屏

HTML5规范允许用户自定义网页上任一元素全屏显示。

requestFullscreen() 开启全屏显示

cancleFullscreen() 关闭全屏显示

不同浏览器需要添加前缀如：

webkitRequestFullScreen、mozRequestFullScreen

webkitCancleFullScreen、mozCancleFullScreen

通过document.fullScreen检测当前是否处于全屏

不同浏览器需要添加前缀

document.webkitIsFullScreen、document.mozFullScreen

全屏伪类

:full-screen .box {}、:-webkit-full-screen {}、:moz-full-screen {}

**var** docElm = **document**.**documentElement**;  
 **if** (docElm.requestFullscreen) {  
 docElm.requestFullscreen();  
 }  
 **else if** (docElm.mozRequestFullScreen) {  
 docElm.mozRequestFullScreen();  
 }  
 **else if** (docElm.webkitRequestFullScreen) {  
 docElm.webkitRequestFullScreen();  
 }

# 十一 网络状态

我们可以通过window.onLine来检测，用户当前的网络状况，返回一个布尔值

window.online用户网络连接时被调用

window.offline用户网络断开时被调用

**window**.addEventListener(**"online"**,**function**(){  
 alert(**"已经建立了网络连接"**)  
 })  
 **window**.addEventListener(**"offline"**,**function**(){  
 alert(**"已经失去了网络连接"**)  
 })

# 十二 应用缓存（离线存储）

HTML5中我们可以轻松的构建一个离线（无网络状态）应用，只需要创建一个cache manifest文件。这样做的优点有：

1、可配置需要缓存的资源

2、网络无连接应用仍可用

3、本地读取缓存资源，提升访问速度，增强用户体验

4、减少请求，缓解服务器负担

## 1 缓存清单

一个普通文本文件，其中列出了浏览器应缓存以供离线访问的资源，推荐使用.appcache为后缀名，添加MIME类型

AddType text/cache-manifest .appcache

例如我们创建了一个名为demo.appcache的文件，然后在需要应用缓存在页面的根元素(html)添加属性manifest="demo.appcache"，路径要保证正确。

## 2 manifest文件格式

1、顶行写CACHE MANIFEST

2、CACHE: 换行 指定我们需要缓存的静态资源，如.css、image、js等

3、NETWORK: 换行 指定需要在线访问的资源，可使用通配符

4、FALLBACK: 当前页面无法访问时退回的页面(回退;  后退)

换行 当被缓存的文件找不到时的备用资源 可自行查阅资料

|  |
| --- |
| CACHE MAINFEST  #要缓存的文件  CACHE:  <http://i.temp.com/images/1.jpg>  [http://i.temp.com/images/2.jpg](http://i.temp.com/images/1.jpg)  images/22.jpg  images/33.jpg  #必须联网才能访问的文件  NETWORK:  <http://www.temp.js.com/jquery1.7.1>  #访问不到某个资源，自动替换,访问不到1.css，自动替换为2.css  FALLBACK:  1.css 2.css |

注意：

CACHE可以省略，这种情况下将需要缓存的资源写在CACHE MANIFEST；

可以指定多个CACHE: NETWORK: FALLBACK:，无顺序限制

chrome 可以通过chrome://appcache-internals/工具和离线（offline）模式来调试管理应用缓存

# 十三 Font Awesome 字体框架

<http://fontawesome.dashgame.com/>

Font Awesome为您提供可缩放的矢量图标，您可以使用CSS所提供的所有特性对它们进行更改，包括：大小、颜色、阴影或者其它任何支持的效果