# 五 新增DOM相关

## 1 新增权责器

document.getElementsByClassName ('class')

通过类名获取元素，以类数组形式存在。

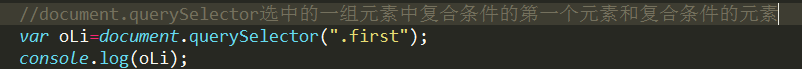
document.querySelector(‘div’)

通过CSS选择器获取元素，符合匹配条件的第1个元素。

document.querySelectorAll('selector')

通过CSS选择器获取元素，以类数组(伪数组)形式存在。

document.querySelector()查询复合该选择器的单个元素





## 2 类名操作

Node.classList.add('class') 添加class

Node.classList.remove('class') 移除class

Node.classList.toggle('class') 切换class，有则移除，无则添加

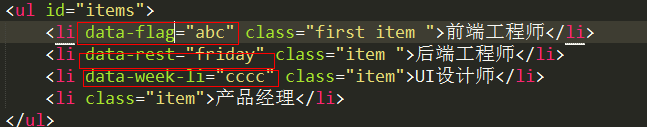
Node.classList.contains('class') 检测是否存在class

Node指一个有效的DOM节点，是一个通称。

## 3 自定义属性data-

在HTML5中我们可以自定义属性，其格式如下data-\*=""，例如：

data-info="我是自定义属性"



通过Node.dataset['info'] 获取到自定义的属性值。

Node.dataset是以类对象形式存在的

当我们如下格式设置时，则需要以驼峰格式才能正确获取

data-my-name="mine"，获取Node.dataset['myName']



## 4 JSON对象新方法

json与字符串的转换我们之前一直使用的是eval方法，在H5中增加了以下两个方法：

parse()： 把类似json的字符串转换成json，字符串中的属性要加上引号；

stringify()： 把json转换成字符串，属性会被自动加上双引号；

eval()： 把任意的字符串转换成JS；

*//eval*var str1 = 'function show(){alert(123)}';  
eval(str1);  
show();  
*//parse*var str2 = '{"age":14}';  
alert(JSON.parse(str2).age);

parse也可以用来拷贝对象，防止对象赋值时出现引用的状况。

## 5 延迟加载属性defer与async

defer：延迟加载，会按顺序执行，在onload执行前触发；

async：异步加载，加载完就触发，有顺序问题。

<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <script defer="defer"></script>  
</head>

在head内的标签由于使用了defer，那么就不会先加载，而是onload之后再加载。注意：defer最好只在加载外部JS时使用，其他场合会有兼容问题。

## 6 拖拽

为元素增加draggable="true"可以设置此元素是否可以进行拖拽。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

**拖拽元素事件（**页面中设置了draggable="true"属性的元素**）**

ondragstart 应用于拖拽元素，当拖拽开始时调用，只执行一次

ondrag 应用于拖拽元素，整个拖拽过程都会调用，一直执行

ondragleave 应用于拖拽元素，当鼠标离开拖拽元素时触发一次

ondragend 应用于拖拽元素，当拖拽结束时调用

**目标元素事件（**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，当拖拽元素进入时触发一次

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

火狐下拖拽对象（dataTransfer）兼容：

ev.dataTransfer.setData(key,value)

ev.dataTransfer.getData(key)

# 六 历史管理

## 1 HTML5新增的历史方法

window.history对象可以管理历史记录，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

新增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

新增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。

## 3 onhashchange()方法

# 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。

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

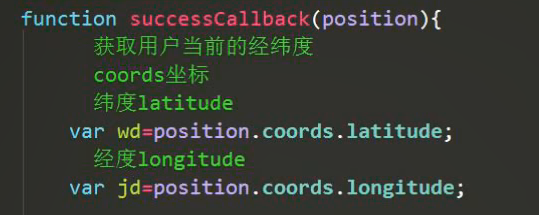
options参数为额外参数，也是可选参数，对象

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。

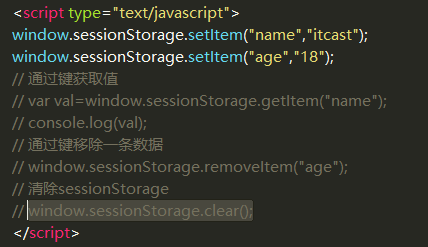
查看方式：Chrome开发者工具中Application

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



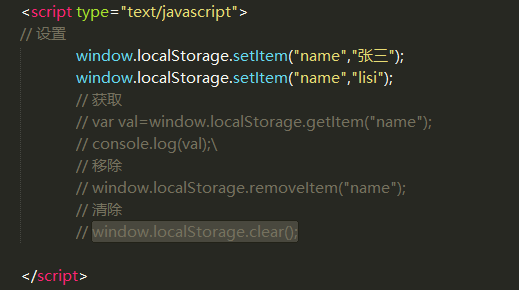
## 2 本地存储

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