一：回顾

二：多媒体：http://www.w3school.com.cn/tags/html\_ref\_audio\_video\_dom.asp

1. 常用方法：load() 加载、 play() 播放、 pause() 暂停

Jq没有提供对视频播放控件的方式，也就意味着如果要操作视频播放，必须使用原生的js方法—dom元素

1. 常用属性：
   1. currentTime 视频播放的当前进度、
   2. duration:视频的总时间 100000/60
   3. paused:视频播放的状态.
2. 常用事件：
   1. oncanplay: 事件在用户可以开始播放视频/音频（audio/video）时触发。
   2. ontimeupdate:通过该事件来报告当前的播放进度.
   3. onended:播放完时触发—重置

三：多媒体案例：自定义视频播放器

1. 效果：



1. 页面结构：

<h3 class="playerTitle">视频播放器</h3>  
<div class="player">  
 <video src="../mp3/test.mp4"></video>  
 <div class="controls">  
 <a href="javascript:;" class="switch fa fa-pause"></a>  
 <a href="javascript:;" class="expand fa fa-expand"></a>  
 <div class="progress">  
 <div class="bar"></div>  
 <div class="loaded"></div>  
 <div class="elapse"></div>  
 </div>  
 <div class="time">  
 <span class="currentTime">00:00:00</span>  
 \  
 <span class="totalTime">00:00:00</span>  
 </div>  
 </div>  
</div>

1. Js代码：

<script src="../js/jquery.min.js"></script>  
<script>  
 /\*获取到播放器\*/  
 **var** video=$("video")[0];  
 /\*暂停-播放切换\*/  
 $(".switch").click(**function**(){  
 //1.切换样式，从暂停切换到播放，或者从播放切换到暂停  
 $(**this**).toggleClass("fa-pause fa-play");  
 //2.修改播放器的状态  
 **if**(video.paused){  
 video.play();  
 }  
 **else**{  
 video.pause();  
 }  
 });  
 /\*全屏\*/  
 $(".expand").click(**function**(){  
 video.webkitRequestFullScreen();  
 });  
 /\*当可以进行播放的时候触发oncanplay\*/  
 video.oncanplay=**function**(){  
 setTimeout(**function**(){  
 video.style.display="block";  
 //1.获取视频的总时长,结果以秒作为单位  
 **var** duration=video.duration;  
 /\*console.log(duration);\*/ //256.278  
 //2.计算 时 分 秒  
 **var** hour= Math.floor(duration/3600);  
 **var** menite= Math.floor(duration%3600/60);  
 **var** second=Math.floor(duration%60);  
 //3.将时分秒信息填充到总时长span中  
 //3.1 设置时分秒的格式  
 hour=hour<10?"0"+hour:hour;  
 menite=menite<10?"0"+menite:menite;  
 second=second<10?"0"+second:second;  
 //3.2填充  
 $(".totalTime").html(hour+":"+menite+":"+second);  
 },2000);  
 }  
 /\*当视频在播放的时候，会触发下下面的方法ontimeupdate\*/  
 video.ontimeupdate=**function**(){  
 //1.获取当前已经播放过了时间  
 **var** elapseTime=video.currentTime;  
 //2.获取已过时间的时分秒  
 **var** hour= Math.floor(elapseTime/3600);  
 **var** menite= Math.floor(elapseTime%3600/60);  
 **var** second=Math.floor(elapseTime%60);  
 //3.将时分秒信息填充到当前时长span中  
 //3.1 设置时分秒的格式  
 hour=hour<10?"0"+hour:hour;  
 menite=menite<10?"0"+menite:menite;  
 second=second<10?"0"+second:second;  
 //3.2填充  
 $(".currentTime").html(hour+":"+menite+":"+second);  
 //4.设置当前<div class="elapse"></div>的宽度  
 **var** valuePercent=0;  
 **if**(elapseTime>0){  
 valuePercent=elapseTime/video.duration\*100;  
 $(".elapse").css("width",valuePercent+"%");  
 }  
 }  
</script>

四：地理定位：了解

1. 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)
2. 获取地理信息方式
   1. IP地址
   2. 三维坐标
      1. GPS（Global Positioning System，全球定位系统）
      2. Wi-Fi
      3. 手机信号
   3. 用户自定义数据

如下图对不同获取方式的优缺点进行了比较，浏览器会自动以最优方式去获取用户地理信息。

1. 隐私: 推送通知

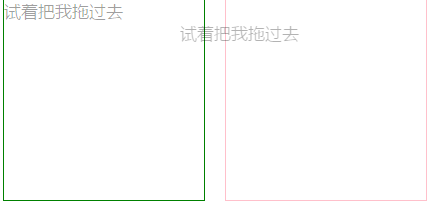
HTML5 Geolocation(地理位置定位) 规范提供了一套保护用户隐私的机制。必须先得到用户明确许可，才能获取用户的位置信息

1. API说明：
   1. navigator.getCurrentPosition(successCallback, errorCallback, options) 获取当前地理信息
   2. navigator.watchPosition(successCallback, errorCallback, options) 重复获取当前地理信息
   3. 1、当成功获取地理信息后，会调用succssCallback，并返回一个包含位置信息的对象position。Coords(坐标)
   4. position.coords.latitude纬度
   5. position.coords.longitude经度
   6. 当获取地理信息失败后，会调用errorCallback，并返回错误信息error
   7. 可选参数 options 对象可以调整位置信息数据收集方式
2. 百度地图案例-介绍

五：拖拽：

## 定义和用法：拖放是 HTML5 中非常常见的功能。

* + 1. 在拖放的过程中会触发以下事件：
       1. 在拖动目标上触发事件 (源元素):
          - [ondragstart](http://www.runoob.com/jsref/event-ondragstart.html) - 用户开始拖动元素时触发
          - ondrag - 元素正在拖动时触发
          - [ondragend](http://www.runoob.com/jsref/event-ondragend.html) - 用户完成元素拖动后触发
       2. 释放目标时触发的事件— 当拖拽元素在目标容器上进行操作的时候:
          - [ondragenter](http://www.runoob.com/jsref/event-ondragenter.html) - 当被鼠标拖动的对象进入其容器范围内时触发此事件
          - [ondragover](http://www.runoob.com/jsref/event-ondragover.html) - 当某被拖动的对象在另一对象容器范围内拖动时触发此事件
          - [ondragleave](http://www.runoob.com/jsref/event-ondragleave.html) - 当被鼠标拖动的对象离开其容器范围内时触发此事件
          - [ondrop](http://www.runoob.com/jsref/event-ondrop.html) - 在一个拖动过程中，释放鼠标键时触发此事件
    2. 注意：
       1. 在拖动元素时，每隔 350 毫秒会触发 ondrag 事件。
       2. 为了让元素可拖动，需要使用 HTML5 [draggable](http://www.runoob.com/tags/att-global-draggable.html) 属性
       3. 链接和图片默认是可拖动的，不需要 draggable 属性
       4. 可以通过addEventListener来添加拖拽相关事件
       5. 事件源：触发事件的源，一般情况下我们会将相同操作的多个对象绑定到同一个处理事件，同时传递当前的对象到处理方法，这就是事件源参数
    3. 小案例：实现元素的拖拽操作：
       1. 效果：



* + - 1. 网页结构代码：

<!--标记是否可以拖动，true表示可以拖动-->  
<div class="div1">  
 <p draggable="true" id="pe">试着把我拖过去</p>  
</div>  
<div class="div2"></div>

* + - 1. Js代码 (几个事件的说明)：

<script>  
 /\*这几次拖拽的事件源都是当前被拖拽的元素\*/  
 /\*开始拖拽\*/  
 document.addEventListener("dragstart",**function**(e){  
 //设置当前目标元素的透明度，产生拖拽效果  
 e.target.style.opacity=0.4;  
 //将当前的被拖拽元素的id号存储到事件源对象中  
 e.dataTransfer.setData("Text", e.target.id);  
 });  
 /\*拖拽进行中\*/  
 document.addEventListener("drag",**function**(e){  
 e.target.parentNode.style.borderColor="green";  
 });  
 /\*拖拽结束\*/  
 document.addEventListener("dragend",**function**(e){  
 e.target.style.opacity=1;  
 e.target.parentNode.style.borderColor="red";  
 });  
  
 /\*下面几个方法的事件源是目标元素，而不是被拖拽的元素\*/  
 /\*当将当前元素拖拽到另外一个元素上时触发\*/  
 document.addEventListener("dragenter",**function**(e){  
 **if**(e.target.className=="div2"){  
 e.target.style.borderColor="pink";  
 }  
 **else if**(e.target.className=="div1"){  
 e.target.style.borderColor="red";  
 }  
 });  
  
 /\*拖拽元素在目标元素上移动时触发\*/  
 document.addEventListener("dragover",**function**(e){  
 /\*默认情况下，一个元素不能拖拽到另外一个元素内，如果想允许拖拽，我们必须阻止默认的事件冒泡\*/  
 e.preventDefault();  
 });  
 /\*当拖拽元素离开目标元素时触发\*/  
 document.addEventListener("dragleave",**function**(e){  
 **if**(e.target.className=="div2"){  
 e.target.style.borderColor="blue";  
 }  
 });  
 /\*当拖拽元素在目标元素上松开的时候触发\*/  
 document.addEventListener("drop",**function**(e){  
 //1.阻止事件冒泡  
 e.preventDefault();  
 **if**(e.target.className=="div2" || e.target.className=="div1"){  
 //2.还原目标元素的默认样式  
 e.target.style.borderColor="blue";  
 //3.获取被拖拽元素的id  
 **var** id= e.dataTransfer.getData("Text");  
 //4.追加被拖拽元素到目标元素  
 e.target.appendChild(document.getElementById(id));  
 }  
 });  
</script>

六：web存储：

1. 需求：随着互联网的快速发展，基于网页的应用越来越普遍，同时也变的越来越复杂，为了满足各种各样的需求，会经常性在本地存储大量的数据，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，给开发带来诸多不便，HTML5规范则提出解决方案
2. HTML5提供的解决方案：
   1. window.sessionStorage
   2. window.localStorage
3. 特点：
4. 设置、读取方便
5. 容量较大，sessionStorage约5M、localStorage约20M
6. 只能存储字符串，可以将对象JSON.stringify() 编码后存储
7. Window.sessionStorage的使用
   1. 特点：
      1. 生命周期为关闭浏览器窗口：相当于存储在当前页面的内内存中
      2. 在同一个窗口下数据可以共享(在当前页面下可以获取到，换另外一个页面下不能获取到)
   2. 方法介绍：(两种存储方式的方法一致)
      1. SetItem(key,value):设置数据，以键值对的方式
      2. getItem(key):通过指定的键获取对应的值内容
      3. removeItem(key):删除指定的key及对应的值内容
      4. clear():清空所有存储内容
   3. 使用说明：

<script>  
 **var** userData=document.getElementById("userName");  
 //存储数据  
 document.getElementById("setData").onclick=**function**(){  
 window.sessionStorage.setItem("userName",userData.value);  
 }  
 //获取数据  
 document.getElementById("getData").onclick=**function**(){  
 **var** value=window.sessionStorage.getItem("userName");  
 alert(value);  
 }  
</script>

1. Window.localStorage的使用
   1. 特点：
      1. 永久生效，除非手动删除：存储在硬盘上
      2. 可以多窗口共享。但是不能跨浏览器
   2. 使用说明：

<script>  
 **var** userData=document.getElementById("userName");  
 //存储数据  
 document.getElementById("setData").onclick=**function**(){  
 window.localStorage.setItem("userName",userData.value);  
 }  
 //获取数据  
 document.getElementById("getData").onclick=**function**(){  
 **var** value=window.localStorage.getItem("userName");  
 alert(value);  
 }  
 //删除数据  
 document.getElementById("removeData").onclick=**function**(){  
 window.localStorage.removeItem("userName");  
 }  
</script>

七：应用缓存：

* + - 1. 概念：使用 HTML5，通过创建 cache manifest 文件，可以轻松地创建 web 应用的离线版本
      2. 优势：
         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
      3. Cache Manifest 基础：

1. 如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：

<!DOCTYPE HTML>

<html manifest="demo.appcache">

...

</html>

1. 每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。
2. manifest 文件的建议的文件扩展名是：".appcache"。
3. 注意，manifest 文件需要配置正确的 MIME-type，即 "text/cache-manifest"。必须在 web 服务器上进行配置
   * + 1. Manifest 文件：
4. 概念：manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）
5. manifest 文件可分为三个部分

* CACHE MANIFEST – 开始
* CACHE在此标题下列出的文件将在首次下载后进行缓存
* NETWORK - 在此标题下列出的文件需要与服务器的连接，且不会被缓存
* FALLBACK - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

1. CACHE MANIFEST说明：

CACHE MANIFEST，放置在第一行，是必需的：

----------------------------------------------------------------------------------------------------

CACHE ：

/theme.css

/logo.gif

/main.js

----------------------------------------------------------------------------------------------------

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的

1. NETWORK说明：

NETWORK 小节规定文件 "login.asp" 永远不会被缓存，且离线时是不可用的，如

-----------------------------------------------------------------------------------------------------

NETWORK:

login.asp

----------------------------------------------------------------------------------------------------

1. FALLBACK说明：

FALLBACK 小节规定如果无法建立因特网连接，就使用指定的资源代替所请求的url的资源，如：

------------------------------------------------------------------------------------------------------

FALLBACK:

/html5/ /404.html

-------------------------------------------------------------------------------------------------------

注释：当html5资源在离线状态下无法请求的时候，就使用404.html代替

1. 其它：

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

2、可以指定多个CACHE: NETWORK: FALLBACK:，无顺序限制

3、#表示注释，只有当demo.appcache文件内容发生改变时或者手动清除缓存后，才会重新缓存。

4、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

1. 更新缓存：

一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使修改了服务器上的文件。为了确保浏览器更新缓存，也需要更新 manifest 文件，也就意味着一旦应用被缓存，它就会保持缓存直到发生下列情况：

1. 用户清空浏览器缓存
2. manifest 文件被修改（参阅下面的提示）
3. 由程序来更新应用缓存

说明：更新注释行中的日期和版本号是一种使浏览器重新缓存文件的办法

八：网络状态

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

1. window.online用户网络连接时被调用
2. window.offline用户网络断开时被调用
3. 示例：

/\*网络连接时调用\*/  
window.addEventListener("online",**function**(e){  
 console.log("ok");  
 console.log(e.returnValue);  
});  
/\*网络断开时调用\*/  
window.addEventListener("offline",**function**(e){  
 console.log("no");  
 console.log(e.returnValue);  
});

九：全屏

btn.onclick=**function**() {  
 /\*能力测试\*/  
 **if**(docuEle.requestFullScreen){  
 document.getElementById("img").requestFullScreen();  
 }  
 **else if**(docuEle.webkitRequestFullScreen){  
 document.getElementById("img").webkitRequestFullScreen();  
 }  
 **else if**(docuEle.mozRequestFullScreen){  
 document.getElementById("img").mozRequestFullScreen();  
 }  
}