昨日知识点回顾：

01、HTML5的介绍及兼容性处理

02、HTML5表单新增的type属性

03、HTML5音频和视频标签的使用

04、HTML5获取dom元素及自定义属性

今日知识点：

01、网络状态改变事件

02、全屏API 实现 元素全屏效果

03、文件读取接口 实现 文件读取预览效果

04、拖拽接口 实现 常见拖拽效果

05、地理定位接口 获取 用户位置信息

06、Web存储接口 实现数据的读写

07、多媒体接口 实现 自定义播放器

# 1 网络监听接口

## 1.1 01-HTML5-网络监听接口

1. **online & offline事件的监听使用**
   1. online:有网络的时候触发
   2. offline没有网络的时候触发

# 2 全屏接口的使用

## 2.1 02-HTML5-全屏接口的使用

1. **开启全屏和退出全屏方法如何调用和兼容性处理？**
   1. **开启全屏：ruquestFullScreen()** 
      1. **谷歌：webkitRuquestFullScreen()**
      2. **火狐：mozRuquestFullScreen()**
      3. **微软IE：msRuquestFullScreen()**

**b) 退出全屏：cancelFullScreen() (兼容性跟开启全屏一样设置)**

1. **退出全屏和是否是全屏状态 针对的对象是？**
   1. **cancelFullScreen() & FullscreenElement 都是对document进行判断**

document.querySelector("#cancelFull").onclick=**function**(){  
 **if**(document.cancelFullScreen){  
 document.cancelFullScreen();  
 }  
 **else if**(document.webkitCancelFullScreen){  
 document.webkitCancelFullScreen();  
 }  
 **else if**(document.mozCancelFullScreen){  
 document.mozCancelFullScreen();  
 }  
 **else if**(document.msCancelFullScreen){  
 document.msCancelFullScreen();  
 }  
}

* 1. **FullscreenElement 中screen s不需要驼峰写法**

document.querySelector("#isFull").onclick=**function**(){  
 **if**(document.fullscreenElement || document.webkitFullscreenElement || document.mozFullScreenElement || document.msFullscreenElement){  
 alert(**true**);  
 }**else**{  
 alert(**false**);  
 }  
}

# 3 FileReader的使用

## 3.1 03-HTML5-FileReader的使用

**步骤分析：**

1、用了一个onchange()事件，只要对象发生改变就触发这个事件

1. 创建一个文件读取对象

var reader=new FileReader();

1. 获取选中的文件 (文件存取在type=”file”表单元素files文件中，且文件是一个数组)，存取在变量file中

var file=document.querySelector("#myFile").files;

1. 读取文件 通过readerAsDataURL()来获取,

**读取后会把文件存在文件读取对象的result中，且是转码的**

**reader.readAsDataURL(file[0]);**

1. 通过事件来捕获传给img对象的src

**onabort:读取文件中断片时触发  
 onerror:读取错误时触发  
 onload:文件读取成功完成时触发  
 onloadend:读取完成时触发，无论成功还是失败  
 onloadstart:开始读取时触发  
 onprogress:读取文件过程中持续触发**

**reader.onload=function(){  
 //console.log(reader.result); 读取的文件在读取对象的reader.result中  
 document.querySelector("img").src=reader.result;  
}  
reader.onprogress=function(e){  
 var percent= e.loaded/ e.total\*100+"%";  
 div.style.width=percent;  
}**

# 4 拖拽接口使用

## 4.1 04-HTML5-拖拽接口使用(上)

1. **用于被拖拽元素事件有哪几种？**

**要拖拽必须先开启拖拽**

**用于被拖拽元素的事件**

**ondrag 应用于拖拽元素，整个拖拽过程都会调用--持续  
 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**

1. **用于目标元素事件有哪几种？**

**应用于目标元素的事件**

**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--持续  
 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**

1. **如何触发ondrap事件以及拖拽元素？**
   1. **由于浏览器会阻止触发ondrap事件，需要在ondrapover事件中阻止浏览器默认行为**
   2. **触发ondrap事件在目标中添加拖拽的对象**

**div2.ondragover=function(e){  
 e.preventDefault();  
}  
div2.ondrop=function(){  
 div2.appendChild(p);  
}**

## 4.2 05-HTML5-拖拽接口使用(中)

1. **任意拖动对象是怎么获取的？**

**通过事件源e.target对象获取**

**2、步骤分析：**

1. 申明一个全局的被拖动对象为null
2. 在拖拽开始的时候获取拖拽对象 存到obj=e.target中
3. 在拖拽事件(ondrop)鼠标离开的时候触发，获取此时的事件源，在给其添加初始触发拖拽的对象obj

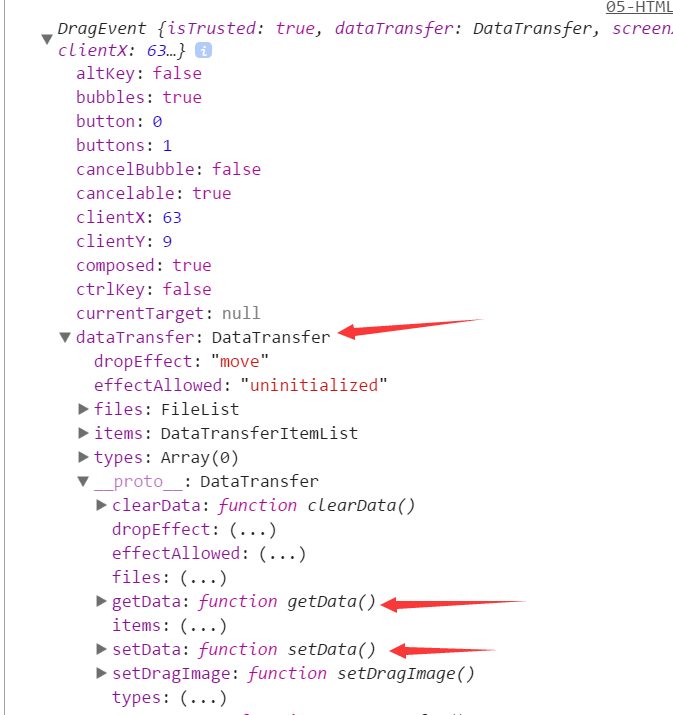
**3、全局变量有哪些缺点？**

* 1. 任何人都可以使用不安全
  2. 全局变量太多会造成内存泄漏

## 4.3 05-HTML5-拖拽接口使用(下)

**1、e.dataTransfer.setData(format,data)的了解**

**通过dataTransfer来实现数据的存储与获取  
 setData(format,data):  
 format:数据的类型：text/html text/uri-list  
 Data:数据:一般来说是字符串值\*/  
e.dataTransfer.setData("text/html", e.target.id);**



1. **e.dataTransfer.setData(format,data)来获取当前拖拽的对象的数据** 
   1. **在开始拖拽就通过e.dataTransfer.setData存储数据 (类型是字符串)**

e.dataTransfer.setData("text/html", e.target.id);

* 1. **在通过e.dataTransfer.getData得到存储的数据 ，在ondrop事件中获取**

document.ondrop=**function**(e){  
 通过e.dataTransfer.setData存储的数据，只能在drop事件中获取  
 **var id=e.dataTransfer.getData("text/html");  
 console.log("id="+id);  
 e.target.appendChild(document.getElementById(id));**  
}

1. **步骤分析：**

**a) 被拖拽时：在ondragstart()获取拖拽对象**

* + 1. 设置当前对象的opacity和父容器的边框
    2. 存储拖拽对象 e.dataTransFer.setData(‘text/html’, e.target.id)

**b) 拖拽至目标对象ondrop()时：**

i. 在ondragover() 阻止浏览器默认行为 e.preventDefault()

ii. 在ondrop() 获取当前e.target , 获取被拖拽时对象的id ， id=e.dataTransFer.getData(‘text/html’);

iii. e.target对象中添加元素

e.target.appengChild(document.getelementById(id))

# 5地理定位接口

## 5.1 06-HTML5-地理定位接口(上)

1. **获取位置信息有哪几种方式？**



1. **地理定位的接口**
   1. **判断地理定位是否存在**
   2. **获取地理信息回调的几种方式**
      1. 获取成功之后的回调
      2. 获取失败之后的回调
      3. 获取调整当前地理信息的方式

**function** getLocation(){  
 **if** (navigator.geolocation){  
 **navigator.geolocation.getCurrentPosition(success,error,option);**

**\* 1.获取地理信息成功之后的回调  
 \* 2.获取地理信息失败之后的回调  
 \* 3.调整获取当前地理信息的方式**  
 **\* option:可以设置获取数据的方式  
 \* enableHighAccuracy:true/false:是否使用高精度  
 \* timeout:设置超时时间，单位ms**

**maximumAge:可以设置浏览器重新获取地理信息的时间间隔，单位是ms** navigator.geolocation.getCurrentPosition(showPosition,showError,{  
 enableHighAccuracy:true,  
 timeout:3000

});  
}**else**{  
 x.innerHTML="Geolocation is not supported by this browser.";  
 }  
}  
  **成功获取定位之后的回调**

如果获取地理信息成功，会将获取到的地理信息传递给成功之后的回调  
 **position.coords.latitude 纬度**

**position.coords.longitude 经度  
 position.coords.accuracy 精度  
 position.coords.altitude 海拔高度**  
**function** showPosition(position){  
 x.innerHTML="Latitude: " + position.coords.latitude +  
 "<br />Longitude: " + position.coords.longitude;  
}  
 **获取定位失败之后的回调**   
**function** showError(error){  
 **switch**(error.code){  
 **case** error.PERMISSION\_DENIED:  
 x.innerHTML="User denied the request for Geolocation."  
 **break**;  
 **case** error.POSITION\_UNAVAILABLE:  
 x.innerHTML="Location information is unavailable."  
 **break**;  
 **case** error.TIMEOUT:  
 x.innerHTML="The request to get user location timed out."  
 **break**;  
 **case** error.UNKNOWN\_ERROR:  
 x.innerHTML="An unknown error occurred."  
 **break**;  
 }  
}  
getLocation();

**一般这样获取比较的麻烦，所以我们都是用第三方地图直接获取，**

**第三方地图类似有：百度地图，高德地图**

## 5.2 07-HTML5-地理定位接口(下)

1. **百度地图使用方法**
   1. 申请密钥
   2. 找到开发，选择javascript API
   3. 找到需要的demo，把代码引入，更换成自己的密钥

<script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak= **DarF2LCCGzn6T16zgy8ZPkvYYE5CT6fu**"></script> **这里的ak值就是要填写密钥**

* 1. 可以更改经纬度来需要的定位

# 6 Web存储接口 & 应用缓存接口

## 6.1 08-HTML5-web存储\_sessionStorage

1. **sessionStorage的使用**

**sessionStorage的使用：**

**1.存储数据到本地。存储的容量5mb左右。  
 2.这个数据本质是存储在当前页面的内存中-意味着其它页面和浏览器无法获取数据  
 3.它的生命周期为关闭当前页面，关闭页面，数据会自动清除  
setItem(key,value):存储数据，以键值对的方式存储  
getItem(key):获取数据，通过指定名称的key获取对应的value值  
removeItem(key):删除数据，通过指定名称key删除对应的值  
clear():清空所有存储的内容**

## 6.2 09-HTML5-web存储\_localStorage

**1、localStorage的使用**

**localStorage的使用：  
 1.存储的内容大概20mb  
 2.不同浏览器不能共享数据,但是在同一个浏览器的不同窗口中可以共享数据  
 3.永久生效，它的数据是存储在硬盘上，并不会随着页面或者浏览器的关闭而清除.如果想清除，必须手动清除  
setItem(key,value):存储数据，以键值对的方式存储  
getItem(key):获取数据，通过指定名称的key获取对应的value值  
removeItem(key):删除数据，通过指定名称key删除对应的值  
clear():清空所有存储的内容**

## 6.3 10-HTML5-应用程序缓存

**1、做应用缓存有哪些好处？**

A) 可配置需要缓存的资源

B) 网络无连接应用仍可用

C) 本地读取缓存资源，提升访问速度，增强用户体验

D) 减少请求，缓解服务器负担

**2、manifest使用说明？**

a) 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>

b) 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）。

c) manifest 文件的建议的文件扩展名是：".appcache"。

d) 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

d) manifest 文件可分为三个部分

i、**CACHE MANIFEST** – 开始

ii、**CACHE**在此标题下列出的文件将在首次下载后进行缓存

Iii、**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

iv、**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）

**3、manifest介绍说明：**

a) 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

b) 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

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

c) 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**

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

# 7多媒体接口

## 7.1 10-HTML5-自定义播放器(接口说明与界面说明)

**1、常用的一些事件、方法、属性有哪些？**

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
3. **底部按钮及进度条的结构布局**



## 7.2 11-HTML5-自定义播放器(播放与暂停)

**步骤分析：**

1. 首先获取播放对象
2. 按钮点击的播放与暂停
   1. 如果视频是暂停就让其播放:video.play() ,否则就暂停:video.pause();
   2. 播放按钮的切换，播放就显示播放按钮图标；暂停显示暂停按钮图标图标，通过切换类名来实现

$(".switch").click(**function**(){  
 实现播放与暂停的切换：如果是暂停>>播放 ，如果是播放 >> 暂停  
 **if**(video.paused){  
 video.play();移除暂停样式，添加播放样式

}**else**{  
 video.pause();移除播放样式，添加暂停样式  
 }  
 设置标签的样式fa-pause:暂停 fa-play：播放  
 $(**this**).toggleClass(" fa-pause fa-play");  
});

1. 全屏操作
   1. 点击通过requestFullScreen()来实现全屏切换 (兼容性问题需要判断)

$(".expand").click(**function**(){  
 /\*全屏>>不同浏览器需要添加不同的前缀>>能力测试\*/  
 **if**(video.requestFullScreen){  
 video.requestFullScreen();  
 }**else if**(video.webkitRequestFullScreen){  
 video.webkitRequestFullScreen();  
 }**else if**(video.mozRequestFullScreen){  
 video.mozRequestFullScreen();  
 }**else if**(video.msRequestFullScreen){  
 video.msRequestFullScreen();  
 }  
});

## 7.3 12-HTML5-自定义播放器(播放过程业务逻辑处理)

**步骤分析：**

1. **视频可以播放触发事件 oncanplay**
   1. 为了过渡到视频显示有动画效果，都加载在定时器中
   2. 显示视频文件
   3. 获取视频的总时长 video.duration得到总时长
   4. 计算时分秒，不足10的在前面补0
   5. 将结果展示在指定的dom元素中

video.oncanplay=**function**(){  
 setTimeout(**function**(){  
 **1.将视频文件设置为显示**  
 **video.style.display="block";**  
 **2.获取当前视频文件的总时长(以秒做为单位，同时获取了小数值)，计算出时分秒** var total=video.duration; //01:01:40 00:00:36  
 3.计算时分少 60\*60=3600  
 \* 3700:3700/3600  
 \* 3700:3700%3600 >> 100 /60  
 **var hour=Math.floor(total/3600);**  
 **4.补0操作**  
 **hour=hour<10?"0"+hour:hour;  
 var minute=Math.floor(total%3600/60);  
 minute=minute<10?"0"+minute:minute;  
 var second=Math.floor(total%60);  
 second=second<10?"0"+second:second;**  
 **5.将计算结果展示在指定的dom元素中**  
 $(".totalTime").html(hour+":"+minute+":"+second);  
 },2000);  
}

**2、视频可以播放触发事件 ontimeupdate (只要时间变化就会触发这个事件)**

**步骤分析：**

1. 获取当前的播放时间 (video.currentTime)
2. 根据当前时间计算出时分秒 (同上面计算方法一样)
3. 将时间展示在指定dom中
4. 封装计算时分秒的方法函数
5. 设置当前播放进度条样式 宽度(百分比)=当前时间/总时长 \*100%

计算时分秒函数封装：

**function** getResult(time){  
 **var** hour=Math.floor(time/3600);  
 /\*补0操作\*/  
 hour=hour<10?"0"+hour:hour;  
 **var** minute=Math.floor(time%3600/60);  
 minute=minute<10?"0"+minute:minute;  
 **var** second=Math.floor(time%60);  
 second=second<10?"0"+second:second;  
 /\***返回结果**\*/  
 **return** hour+":"+minute+":"+second;   
}

封装前：

video.ontimeupdate=**function**(){  
 /\*1.获取当前的播放时间\*/  
 **var** current=video.currentTime;  
 /\*2.计算出时分秒\*/

**var** hour=Math.floor(time/3600);  
 /\*补0操作\*/  
 hour=hour<10?"0"+hour:hour;  
 **var** minute=Math.floor(time%3600/60);  
 minute=minute<10?"0"+minute:minute;  
 **var** second=Math.floor(time%60);  
 second=second<10?"0"+second:second;  
 /\*3.将结果展示在指定的dom元素中\*/  
 $(".currentTime").html(hour+":"+minute+":"+second);  
 /\*4.设置当前播放进度条样式 0.12>>0.12\*100=12+%>12%\*/  
 **var** percent=current/video.duration\*100+"%";  
 $(".elapse").css("width",percent);  
}

封装后：

video.ontimeupdate=**function**(){  
 /\*1.获取当前的播放时间\*/  
 **var** current=video.currentTime;  
 /\*2.计算出时分秒\*/

**var result=**getResult(current);  
 /\*3.将结果展示在指定的dom元素中\*/  
 $(".currentTime").html(**result**);  
 /\*4.设置当前播放进度条样式 0.12>>0.12\*100=12+%>12%\*/  
 **var** percent=current/video.duration\*100+"%";  
 $(".elapse").css("width",percent);  
}

## 7.4 13-HTML5-自定义播放器(跳播和播放结束业务处理)

1. **视频如何实现跳转？**
   1. 点击$(“.bar”) 通过事件对象来获取当前点击处相对于父元素的位偏移

**var** offset= e.offsetX;

* 1. 获取当前位偏移相对于父元素总宽度的百分比

**var** percent=offset/$(**this**).width();

* 1. 计算当前位置占总时长的值(即进度值)

**var** current=percent\*video.duration;

* 1. 当前视频播放时长=进度值

video.currentTime=current;

**注意：当我们改变current就会触发ontimeupdate这个事件，就会执行事件里面的代码 也就改变了当前的进度条，所以我们点击的时候只需要获取当前所点击处的当前时长就可以实现视频跳转 以本地文件打开即可，webstorm不支持**

1. **播放完毕回到重置状态的设置**
   1. 播放完毕后会触发 onended事件
   2. 初始化当前时间，讲暂停按钮类名切换成播放按钮

video.onended=**function**(){  
 video.currentTime=0;  
 $(".switch").removeClass("fa-pause").addClass("fa-play");  
}