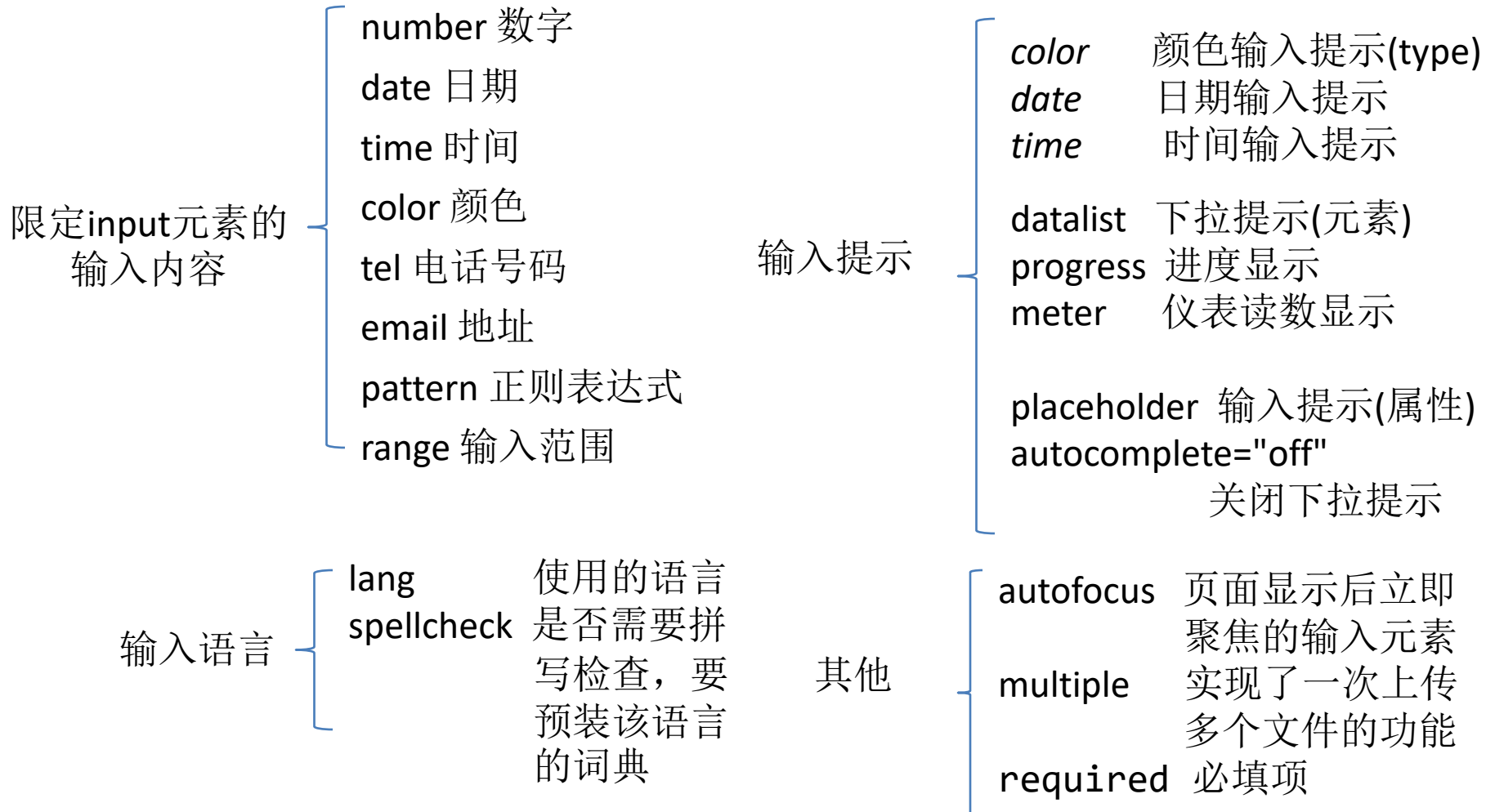


HTML5

2017.12.26

isszym sysu.edu.cn

输入元素



参考源码: <http://172.18.187.11:8080/html5/>

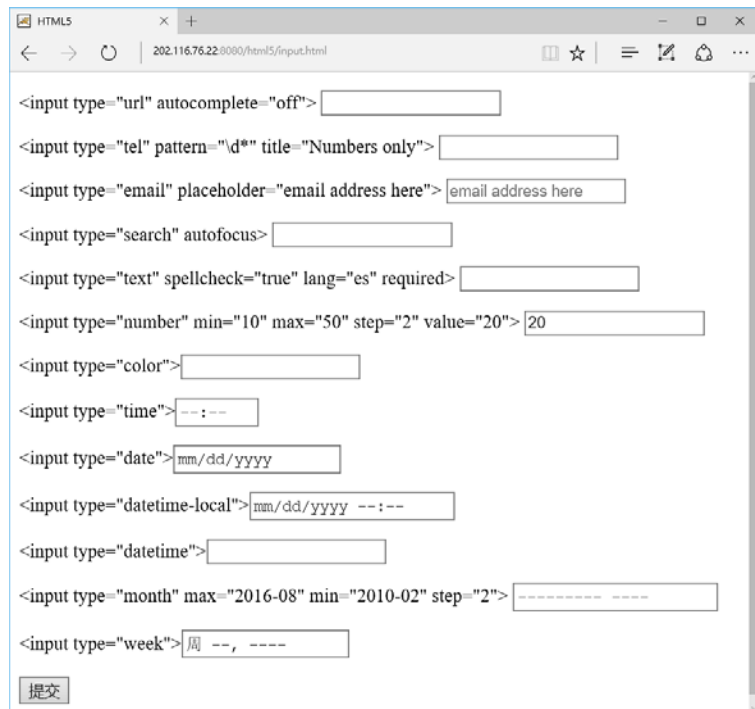
```
<!DOCTYPE html><html><head>
  <title>HTML5</title><meta char="utf-8"></head>
<body>
  <form action="html5.jsp" method="post">
    <input type="url" autocomplete="off">
    <input type="tel" pattern="\d*" title="Numbers only">
    <input type="email" placeholder="input email address here">
    <input type="search" autofocus>
    <input type="text" spellcheck="true" lang="en" required>
    <input type="number" min="10" max="50" step="2" value="20">
    <input type="color">
    <input type="time">
    <input type="date">
    <input type="datetime-local">
    <input type="datetime">
    <input type="month" max="2016-08"
      min="2010-02" step="2">
    <input type="week">
    <input type="submit" value="提交" />
  </form>
</body>
</html>
```

* search对输入没有限制，有的浏览器会用圆角框显示文本框。

* date可以直接取到date类型的值：

```
var date1=document.querySelector("input[type=date]");
```

```
var date2=date1.valueAsDate; //valueAsNumber 对于number类型
```



```
<!DOCTYPE html><html><head>
<title>HTML5</title>
<meta char="utf-8">
<script>
function rangeChange(){
    var rangeText=document.getElementById("rangeText");
    var output=document.getElementById("output");
    rangeText.value=event.currentTarget.value;
    output.value=rangeText.value;
    output.innerHTML = rangeText.value;
}
</script></head><body>
<form action="html5.jsp" method="post">
    <input type="range" min="10" max="50" step="2" value="20"
        onchange="rangeChange()" id="range">
    <input type="text" readonly id="rangeText" value="20">
    <output id="output" for="range">20</output>
    <progress min="10" max="50" value="20">20</progress>
    <input type="text" list="quality">
    <datalist id="quality">
        <option>First Class</option>
        <option>Second Class</option>
        <option>Third Class</option>
        <option>Fourth Class</option>
        <option>Fifth Class</option>
    </datalist>
```

```

<meter low="0.2" high="0.8" value="0.85">0.85 of 1 </meter>
<meter low="0.2" high="0.8" optimum="0.9" value="0.85">0.85 of 1</meter>
<meter low="0.2" high="0.8" optimum="0.1" value="0.2">0.85 of 1 </meter>
<input id="frm1" type="submit" value="提交" />
<input type="file" multiple form="frm1" style="width:500px">
</frm>
</body>
</html>

```

- * 可以把要提交的元素放在form外边，用属性form进行关联
- * range可以利用onchange事件得到当前value
- * range可以采用以下语句减少值：
var rng=document.querySelector("input[type=range]");
rng.stepDown(2);
- * progress的属性value与进度关联，属性position也与进度关联，只是position取值0~1。
- * output的属性defaultValue用于设置默认值。
- * **<button>** 元素在HTML 5 中增加的类似form元素中使用的属性： formaction, formenctype, formmethod, formnovalidate 以及 formtarget。 还有属性autofocus和 form。 form可以指定多个要提交的表单名，用空格隔开。

```

<button type="submit" form="frm1 frm2" value="Submit">提交</button>


```

text
output

HTML5


202.116.76.22:8080/html5/input2.html

```
<input type="range" min="10" max="50" step="2" value="20" onchange="rangeChange()" >
```

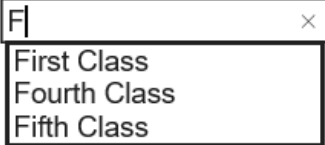


28

```
<progress min="10" max="50" value="20">20</progress>
```



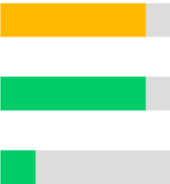
```
<input type="text" list="quality">
```



```
<meter low="0.2" high="0.8" value="0.85">0.85 of 1 </meter>
```

```
<meter low="0.2" high="0.8" optimum="0.9" value="0.85">0.85 of 1 </meter>
```

```
<meter low="0.2" high="0.8" optimum="0.1" value="0.2">0.85 of 1 </meter>
```



提交

- meter的默认最优值为low和high之间，value大于high变黄色，否则为绿色。
- 如果最优值大于high，则全部取值都是绿色。
- 如果最优值小于low，则value只有小于等于low取绿色，其它值都为黄色。

客户端表单验证

- 自动验证

- 输入元素在提交时进行合法性验证。如果不为空且不符合限定条件，则会产生错误提示。加上属性**required**的输入元素要求必须有输入值，否则验证时会提示出错，对于值为空且没有加上**required**的输入元素，不进行验证。具有**disabled**属性或者**formnovalidate**属性的元素也不参与验证。
- 如果要对整个表单禁用提交验证，可以在表单元素中加入属性**novalidate**:
`<form action="post.jsp" method="post" novalidate>`

- 验证方法

方法1：利用form的submit事件进行验证。

方法2：利用元素的input事件和checkValidity()进行验证。

方法3：利用input事件和setCustomValidity()进行验证。

方法4：利用invalid事件捕捉输入错误。

方法1: 利用form的submit事件进行验证。对于每个input元素采用willValidate属性判断是否需要验证，如果需要则采用checkValidity()进行验证，没有通过验证则返回false取消提交操作。

[validate1](#)

```
<!DOCTYPE html><html><head><title>CheckValid-method 1</title>
<meta char="utf-8">
<script>
function validate(){
    var inputs = document.querySelectorAll('input');
    for(var i=0;i<inputs.length;i++){
        if(inputs[i].willValidate && !inputs[i].checkValidity()){
            alert(inputs[i].title+":输入错误! ");
            inputs[i].focus();
            return false;
        }
    }
    return true;
}
</script></head><body>
<form action="html5.jsp" method="post" novalidate onsubmit="return
                                                                    validate();">
tel#: <input type="tel" pattern="\d*" title="电话号码"></p>
Type#:<input type="number" min="10" max="50" step="2"
                                             value="20" title="型号"></p>
<p>    <input type="submit" value="提交"/></p>
</form></body></html>
```


方法2： 利用元素的input事件和checkValidity()进行验证。

[validate2](#)

```
<!DOCTYPE html><html><head><title>CheckValid-method 2</title>
<meta char="utf-8">
<script>
function showInvalidMessage(){
    obj=event.currentTarget;
    if(!obj.checkValidity()){
        var status= obj.validity;
        if(status.patternMismatch){ alert("模式不匹配"); }
        else if (status.typeMismatch){ alert("类型不匹配"); }
        else if (status.rangeUnderflow){ alert("值太小"); }
        else if (status.rangeOverflow){ alert("超出范围"); }
        else{ alert("填写错误! "); }
    } //customError,rangeUnderflow, stepMismatch, tooLong, valid, valueMissing
}
</script></head><body>
<form action="html5.jsp" method="post">
tel#: <input type="tel" pattern="\d*" title="电话号码"></p>
Type#:<input type="number" min="10" max="50" step="2"
        value="20" title="型号" oninput="showInvalidMessage()"></p>
<p>    <input type="submit" value="提交"/></p>
</form></body></html>
```

方法3：利用input事件和setCustomValidity()进行验证。

[validate3](#)

```
<!DOCTYPE html><html><head><title>CheckValid-method 3</title>
  <meta char="utf-8">
</head><body>
<form action="html5.jsp" method="post">
<p>密码:<input type="password" id="password"></p>
<p>重复密码:<input type="password" id="repassword"></p>
<p><input type="submit" value="提交"/></p>
</form>
<script>
  var repass = document.getElementById("repassword");
  repass.addEventListener("input",
    function(e){
      var pass = document.getElementById("password");
      if(e.target.value===pass.value){
        e.target.setCustomValidity("");
      }
      else{
        e.target.setCustomValidity("两次输入的密码不同！");
      }
    }, false)
</script></body></html>
```

方法4：利用invalid事件捕捉输入错误。

[validate4](#)

```
<!DOCTYPE html>
<html>
<head>
  <title>CheckValid-method 4</title>
  <meta char="utf-8">
</head>
<body>
  <form action="html5.jsp" method="post">
    <p>编号:<input type="text" id="num" pattern="\d*" title="全数字"></p>
    <p><input type="submit" value="提交"/></p>
  </form>
  <script>
    function logInvalid(){
      alert("要求全数字! ");
    }
    var num = document.getElementById("num");
    num.addEventListener("invalid",logInvalid,false)
  </script>
</body>
</html>
```

表单CSS

- 新增伪类

```
input:required {border-color:blue}  
input:optional {border-color:black}  
input:valid {border-color:black}  
input:invalid {border-color:red}  
input:range {border-color:black}  
input:out-of-range {border-color:yellow}
```

必须输入
可选输入
输入值有效
输入值无效
输入值在范围内
输入值超出范围

- 原有伪类

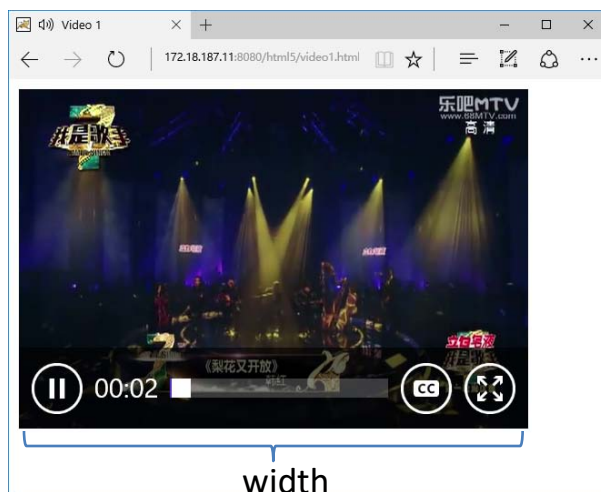
```
.check:indeterminate {border-color:brown} //用于checkbox和radio的第三种状态  
.check:checked {border-color:brown}  
.check:not(:checked) {border-color:brown}  
input:read-only {border-color:brown}  
input:read-write {border-color:brown}  
input:disabled {border-color:brown}  
input:abled {border-color:brown}
```

Video和Audio

- 一般格式

[video1.html](#)

```
<video src="hh.mp4" controls loop autoplay poster="hh.jpg"></video>
<audio src="abc.mp3" controls loop autoplay></audio>
```



- controls 是否显示控制按钮
- loop 是否循环播放
- autoplay 是否立即播放
- muted 是否静音
- poster 非立即播放时显示的图片
- preload 设置媒体文件预下载功能

- preload {
- metadata--只下载文件的元数据(文件大小等)到内存中，而媒体文件要等到播放时下载；
 - auto--浏览器在后台下载。在不是立即播放的情况下，用poster显示一幅图像，以免出现空白画面。

[video2.html](#)

- 部分播放

[video3.html](#)

```
<video src="video/zxz.mp4#4,8" ></video>  
<video src="video/zxz.mp4#9.5," ></video>
```

只播放4到8秒期间的视频
播放9.5秒之后的全部视频

- 视频的字幕显示

采用Web 视频文本轨道文件

[video4.html](#)

```
<video src="video/hh.mp4" style="width:400px" controls autoplay loop>  
  <track kind="subtitles" src="video/hh_cn.vtt"  
    srclang="cn" label="中文" default>  
  <track kind="subtitles" src="video/hh_en.vtt"  
    srclang="en" label="English">  
</video>
```

[hh_cn.vtt](#)

Web视频文本轨道: Web Video Text Track, WebVTT

```
WEBVTT  
1  
00:00:30.500 --> 00:00:49.500  
忘不了故乡 年年梨花放 染白了山冈 我的小村庄  
2  
00:00:50.500 --> 00:01:05.500  
妈妈坐在梨树下 纺车嗡嗡响 我爬上梨树枝 闻那梨花香
```

采用计时文本标记语言：

[video5.html](#)

```
<video src="video/hh.mp4" style="width:800px" controls autoplay loop>
  <track kind="subtitles" src="video/hh_cn.ttml">
  <track kind="subtitles" src="video/hh_en.ttml" default>
</video>
```

[hh_cn.ttml](#)

```
<?xml version='1.0' encoding='UTF-8'?>
<tt xmlns='http://www.w3.org/ns/ttml' xml:lang='en' >
<body>
  <div>
    <p begin="00:00:20.878" end="00:00:49.334">忘不了故乡 年年梨花放<br/>染
      白了山冈 我的小村庄</p>
    <p begin="00:00:50.608" end="00:01:05.296" >妈妈坐在梨树下 纺车嗡嗡响
      <br/>我爬上梨树枝 闻那梨花香</p>
  </div>
</body>
</tt>
```

[计时文本标记语言：Timing Text Markup Language, TTML](#)

- 媒体对象的属性、事件和方法

方法

play 播放
stop 暂停
load 载入

getElementById("video1")

**<video src="video/hh.mp4"
id="video1">**

属性

paused 取到是否暂停的状态
playbackRate 设置快进(0~1)和快退(-1~0)
volume 取得和设置音量大小(0~1)
currentTime 取到或设置当前播放时间(>=0)
duration 取得整个视频的播放时间

事件

playing	首次播放
play	暂停之后重新播放
pause	播放后暂停
abort	因某些原因播放被中断
ended	全部播放完毕
seeking	搜索条被使用
seeked	搜索结束
volumechange	volume属性值变化时或muted改变
timeupdate	currentTime属性更新

- * 对于音频<audio src="video/hh.mp3" id="audio1">, 上面的属性、方法、事件都可用。
- * 当src改变时, 要采用load方法装载新的媒体内容。
- * 视频对象和音频对象的API接口分别为HTMLVideoElement和HTMLAudioElement。

- 属性**readyState**

要获得媒体的元数据可以通过定时器不断获取属性**readyState**，当**readyState**>1时就可以获得**duration**和实际视频的尺寸**videoWidth**和**videoHeight**。

属性**readyState**表示当前的媒体状态：

- 0 无法获得任何媒体状态
- 1 媒体的元数据加载完毕
- 2 可以获得有关当前画面或回放位置的信息
- 3 可以获得当前画面以及至少下一幅画面的信息
- 4 数据充足和下载速度够快从而使媒体可以播放到结尾。

属性**readyState**)的变化会触发以下事件：

- 1 loadedmetadata
- 2 loadeddata
- 3 canplay（可以播放）
- 4 canplaythrough（可以连续播放）。

例 1、播放、暂停、调音量、静音

video6

```
<!DOCTYPE html><html><head><title>Video 6</title><meta char="utf-8"></head>
<body>
  <video id="video" src="video/hh.mp4" style="width:600px"></video>
  <button id="play" title="Play Button">&gt;</button>&nbsp;
  <button id="pause" title="Pause Button">||</button>&nbsp;&nbsp;
  <button id="volumeUp" title="Volume Up">+</button>&nbsp;
  <button id="volumeDown" title="Volume Down">-</button>&nbsp;&nbsp;
  <input type="checkbox" id="muted" title="Muted">muted
  <script>
    var video = document.getElementById("video");
    var btnPlay = document.getElementById("play");
    var btnPause = document.getElementById("pause");
    var btnVolUp = document.getElementById("volumeUp");
    var btnVolDown = document.getElementById("volumeDown");
    var btnMuted = document.getElementById("muted");
    btnPlay.addEventListener("click",function(){video.play();});
    btnPause.addEventListener("click",function(){video.pause();});
    btnVolUp.addEventListener("click",
      function(){video.volume+=0.1;
        if(video.volume>1)video.volume=1;});
    btnVolDown.addEventListener("click",
      function(){video.volume-=0.1;
        if(video.volume<0)video.volume=0;});
    btnMuted.addEventListener("click",
      function(){video.muted=btnMuted.checked});
  </script>
</body>
</html>
```



> || + - ☐ muted

例 2、快进、快退、进度条

[video7](#)

```
<!DOCTYPE html><html><head>
  <title>Video 7</title><meta char="utf-8"></head>
<body>
  <video id="video" src="video/hh.mp4" style="width:400px" autoplay></video>
  <input type="range" min="0" max="100" value="0" step="1" onchange="rangeChange()"
    style="width:600px" id="range">
  <button id="fb" title="Fast Backward">&lt;&lt;</button>
  <button id="ff" title="Fast Forward">&gt;&gt;</button></p>
  <script>
    var video = document.getElementById("video");
    var range = document.getElementById("range");
    var fb = document.getElementById("fb");
    var ff = document.getElementById("ff");
    video.addEventListener("timeupdate",timeupdate);
    fb.addEventListener("click",function()
      {video.playbackRate -= .25;});
    ff.addEventListener("click",function()
      {video.playbackRate += .25;});

    function timeupdate(){
      range.value=(video.currentTime/video.duration)*100;
    }
    function rangeChange(){
      video.currentTime=video.duration*range.value/100;
    }
  </script></body></html>
```



例3、接续播放多个媒体文件

[video8](#)

```
<!DOCTYPE html>
<html>
<head>
<title>Video 6</title>
<meta char="utf-8">
</head>
<body>
<video id="video" src="video/hh.mp4" style="width:600px" autoplay controls>
</video>
<script>
var URL=["video/hh.mp4","video/zxz.mp4","video/wf.mp4"];
var index=0;
var video = document.getElementById("video");
video.addEventListener("ended",
function(){index=(index==2)?0:index+1;video.src=URL[index];video.load()});
</script>
</body>
```



- 多源文件

为了预防浏览器可能不支持某些视频文件，可以采用多源定义方式，还可以加上媒体查询，浏览器将播放符合媒体查询的第一个支持的视频格式：

```
<video>
  <source src="hh.ogv" type="video/ogg"></source>
  <source src="hh.webm" type="video/webm"></source>
  <source src="hh-hd.mp4" type="video/mp4"
    media="(min-device-width:1920px)" ></source>
  <source src="hh.mp4" type="video/mp4" ></source>
</video>
```

如果浏览器不支持媒体元素，为了不显示空白画面，可以加入一幅图像：

```
<video src="hh.mp4">
  
</video>
```

还可以用一个flash播放器作为备用方案，object元素和embed元素可以用flash播放视频，object用于IE浏览器，embed元素用于火狐等浏览器：

```
<video src="hh.mp4">
  <OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    WIDTH="550" HEIGHT="400"
    TYPE="application/x-shockwave-flash">
      <PARAM NAME=movie VALUE="hh.mp4">
      <PARAM NAME=quality VALUE=high>
      <EMBED src="hh.mp4" quality=high
        WIDTH="400" HEIGHT="400" NAME="swf1"

        TYPE="application/x-shockwave-flash"
      </EMBED>
    </OBJECT>
  </video>
```

本地存储

本地存储localStorage可以用来长期存储一些键值对，这个功能与cookie很相似，只是localStorage可以存储更多的数据，cookie的小于3KB，localStorage的小于3MB。localStorage不能跨域访问，没有过期时间。

```
localStorage["author"]="John Waiters";  
localStorage.bookName="Old Days 123";  
localStorage.setItem("year","2016");  
  
alert(localStorage.bookName);  
alert(localStorage.getItem("year"));  
alert(localStorage["author"]);  
  
localStorage.removeItem("year");    // 删除year的键值对  
localStorage.clear();               // 删除所有键值对
```

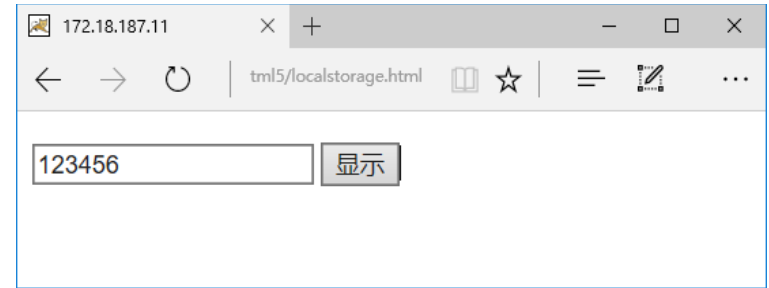
当某个键的值发生改变时会触发storage事件，event对象此时可以取到key、oldValue、newValue。

sessionStorage功能与localStorage的相同，只是sessionStorage不能永久保存键值对，在浏览器完全关闭之后用它保存的键值对就会被清除。sessionStorage可以用于页面之间传递参数。

[参考](#)

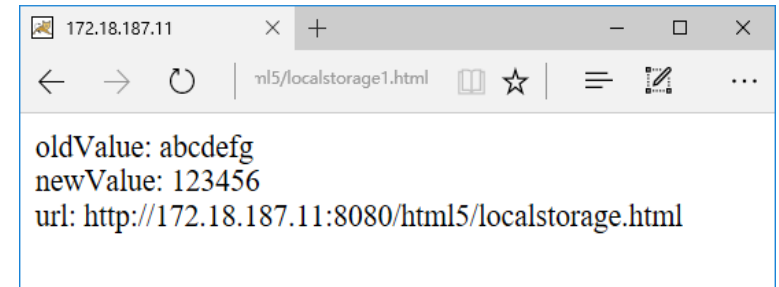
localStorage.html

```
<!DOCTYPE html>
<html> <head> <meta charset="utf-8">
  <title></title>
</head><body>
<p><input id='text' placeholder="保存">
<button id='btn'>显示</button>    </p>
</body></html>
<script>
  var txt = document.getElementById("text")
  var btn = document.getElementById("btn");
  btn.addEventListener('click',function(){
    localStorage.setItem("key", txt.value);
  })
</script>
```



localStorage1.html

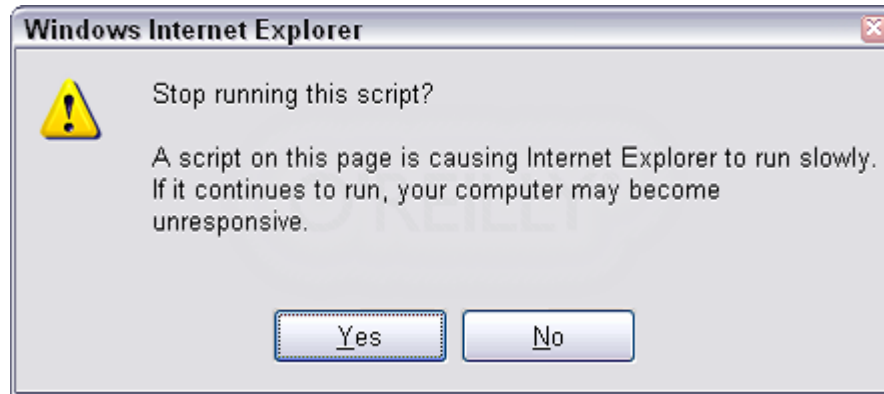
```
<!DOCTYPE html> <html> <head>
  <meta charset="utf-8">
  <title></title>
</head>
<body>
  <script>
    window.addEventListener("storage", function(e){
      document.write("oldValue: " + e.oldValue + "<br>newValue: " + e.newValue
        + "<br>url: " + e.url);
    });
  </script>
</body>
</html>
```



[localStorageRead](#) [localStorageSave](#)
[sessionStorageRead](#) [sessionStorageSave](#)

Worker

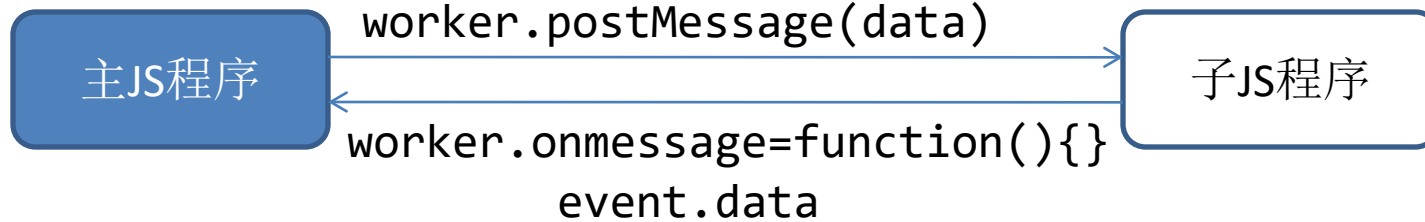
JavaScript采用单线程执行。但是现在Web应用程序把大量的界面处理和数据处理都搬到了前台，再通过单线程的Javascript程序来处理这些任务就显得力不从心了。当Javascript程序运行时间太长，很多浏览器就处于未响应状态，点击时会出现终止执行的选项。



一些开发人员想了一些方法，例如，依靠定时器、AJAX技术和DOM事件来实现并发，Web Workers 是 HTML5 提供的一个javascript多线程解决方案，使我们可以将一些大计算量的代码交由Web Worker运行而不冻结用户界面。

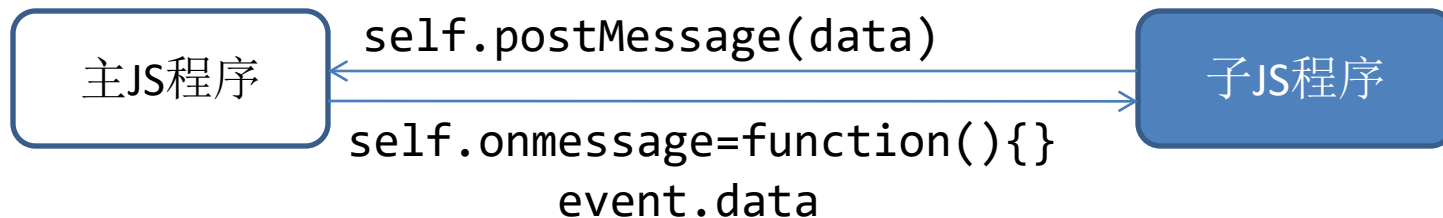
[参考](#)

```
var worker=new Worker("child.js");
```



主Javascript程序：

- (1) 建立 Worker（新的子线程）：`var worker=new Worker("child.js");`
- (2) 子线程数据到达的事件：`worker.onmessage=function(){} event.data`为传送过来的数据（对象）和`event.target`指明哪个Worker（多个Worker使用同一个function）。
- (3) 把数据发送给子线程的方法：`worker.postMessage(data)`，`data`为对象。
- (4) 关闭子线程的方法：`worker.terminate()`



子Javascript程序：

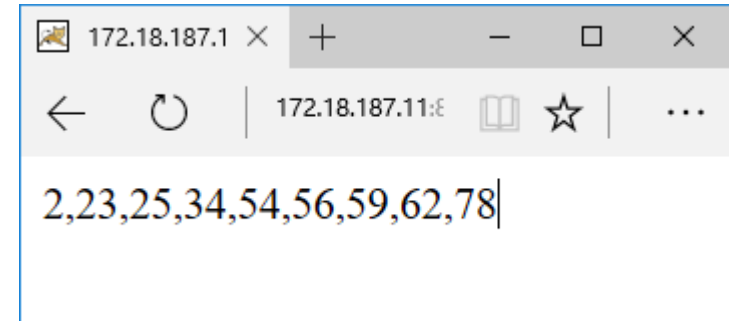
- (1) 用`self`表示自己
- (2) 父线程数据到达的事件：`self.onmessage=function(){} event.data`为传送过来的数据（对象）。
- (3) 把数据发送给父线程的方法：`self.postMessage(data)`，`data`为对象。
- (4) 关闭自己的方法：`self.close()`;

webWorker.html

```
<!DOCTYPE html><html><head>
<title>Web Worker</title>
<meta char="utf-8">
</head>
<body>
<script>
  var data=[34,2,56,23,59,54,62,78,25];
  var worker=new Worker("js/workerExample.js");
  worker.onmessage = function(){
    var data=event.data;
    document.write(data.toString());
  }
  worker.postMessage(data);
</script>
</body></html>
```

workerExample.js

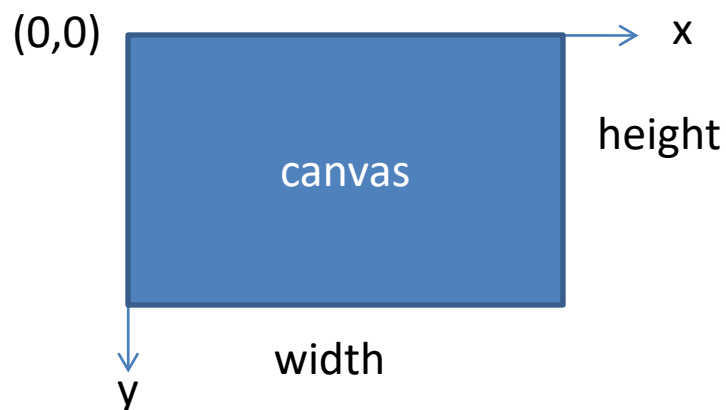
```
self.onmessage=function(event){
  var data=event.data;
  data.sort(function(a,b){
    return a-b;
  });
  self.postMessage(data);
}
```



画布(canvas)

HTML5引入了画布对象，使得在浏览器中可以画图。画布对象绘制的图像是栅格图像，即由点（pixel，像素）构成的图像。

```
<canvas id="canvas" height="400" width="800">
```



用属性定义画布的宽度和高度才能正确作画。

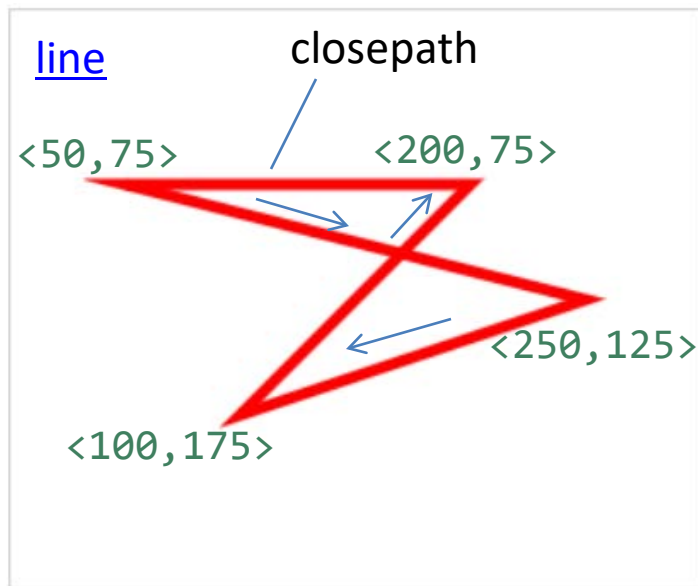
```
var c=document.getElementById("myCanvas");  
var ctx=c.getContext("2d");
```

//用ctx来作画

[参考:W3school](#)
[参考: Context2D 对象](#)
[参考1](#) [2](#) [3](#) [4](#) [5](#)

● 画线

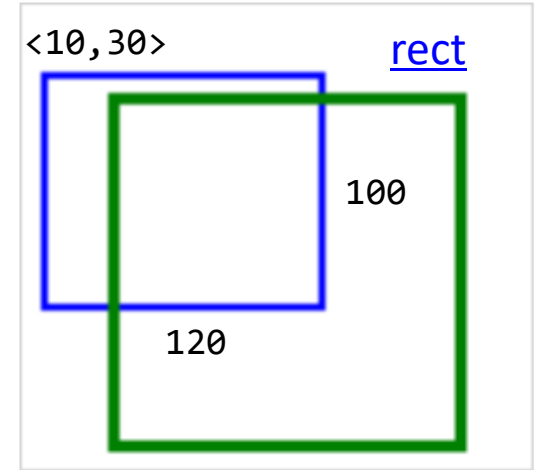
```
<!DOCTYPE html>
<html>
  <body>
    <h1>画线</h1>
    <canvas id="myCanvas" width="300" height="250"
      style="border:1px solid #d3d3d3;">
      Your browser does not support canvas.
    </canvas>
    <script>
      var c=document.getElementById("myCanvas");
      var ctx=c.getContext("2d");
      ctx.beginPath();           // 绘制路径开始
      ctx.lineWidth="5";        // 线宽
      ctx.strokeStyle="#F00";   // 红色（red）路径
      ctx.moveTo(50,75);         // 提笔移至<50,75>并下笔
      ctx.lineTo(250,125);       // 先划线至<250,125>
      ctx.lineTo(100,175);       // 再划线至<100,175>
      ctx.lineTo(200,75);       // 再划线至<200,75>
      ctx.closePath();           // 关闭路径
      ctx.stroke();              // 进行绘制
    </script>
  </body>
</html>
```



● 画矩形

```
<!DOCTYPE html><html>
<body><h1>画矩形</h1>
  <canvas id="myCanvas" width="220" height="200"
    style="border:1px solid #d3d3d3;">
    Your browser does not support canvas.
  </canvas>
  <script>
    var c=document.getElementById("myCanvas");
    var ctx=c.getContext("2d");
    ctx.lineWidth="3";           // 线宽
    ctx.strokeStyle="blue";      // 蓝色线
    ctx.strokeRect(10,30,120,100); // 直接绘矩形: 左上角<x,y>, 宽度, 高度

    ctx.beginPath();             // 绘制路径开始
    ctx.lineWidth="5";           // 线宽
    ctx.strokeStyle="green";     // 绿色线
    ctx.rect(40,40,150,150);     // 画一个矩形
    ctx.stroke();                 // 进行绘制
  </script>
</body></html>
```



● 画弧和圆

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="200"
        style="border:1px solid #d3d3d3;">
```

Your browser does not support the HTML5 canvas tag.

```
</canvas>
```

```
<script>
```

```
var c=document.getElementById("myCanvas");
```

```
var ctx=c.getContext("2d");
```

```
ctx.beginPath();
```

```
ctx.strokeStyle="blue";
```

```
ctx.lineWidth=3;
```

```
ctx.arc(40,10,60,0,Math.PI*2/3,false);//圆心,半径(60),起始角度,结束角度,顺时针方向
```

```
ctx.stroke();
```

```
ctx.beginPath();
```

```
ctx.strokeStyle="red";
```

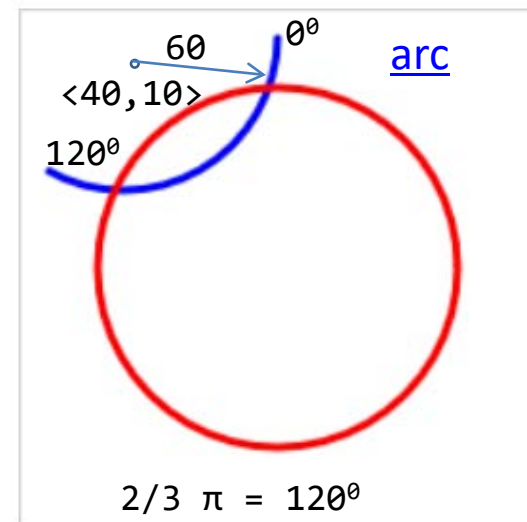
```
ctx.arc(100,75,50,0,2*Math.PI);
```

```
ctx.stroke();
```

```
</script>
```

```
</body>
```

```
</html>
```

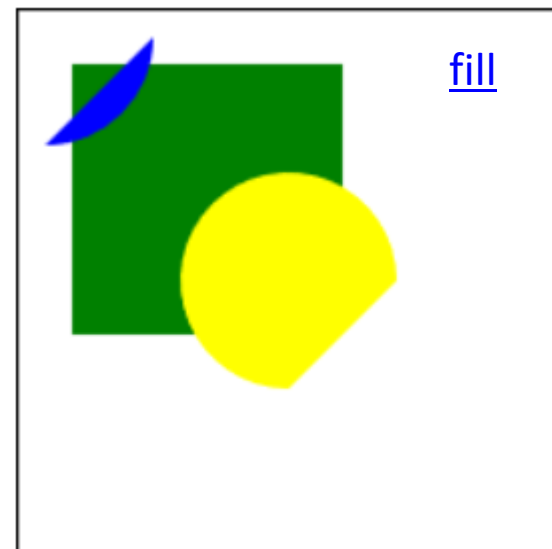


● 画实心图

```
<!DOCTYPE html><html>
<body>
<canvas id="canvas" style="border:solid 1px black;"
        width="200" height="200" >
    canvas is not supported!
</canvas>
<script>
    var can=document.getElementById("canvas");
    var context=canvas.getContext('2d');
    context.fillStyle="green";
    context.fillRect(20,20,100,100);

    context.fillStyle="blue";
    context.arc(10,10,40,0,Math.PI/2,false);
    context.fill();

    context.beginPath();
    context.fillStyle="yellow";
    context.arc(100,100,40,0,Math.PI/2,true);
    context.fill();
</script>
</body></html>
```



• 显示文字

```
<!DOCTYPE html><html>
<body>
  <canvas id="canvas" style="border:solid 1px gray;"
        width="200" height="200" >
    canvas is not supported!
  </canvas>

  <script>
    var can=document.getElementById("canvas");
    var context=canvas.getContext('2d');
    var text = "context";
    context.fillStyle = 'orange';
    context.strokeStyle = 'cornflowerblue';
    context.font = '48px Helvetica';
    context.fillText(text,20,50);      //显示实心字, 起始坐标<20,50>
    context.strokeText(text,20,90);    //显示空心字, 起始坐标<20,90>
    context.fillText(text,20,150);
    context.strokeText(text,20,150);
  </script>
</body></html>
```



● 贴图

```

```

先获得图像对象：

```
var img = document.getElementById("butterfly");
```

然后把图贴到画布上：

```
// 方法1、直接把图贴到位置<x,y>处
```

```
context.drawImage(img,x,y);
```

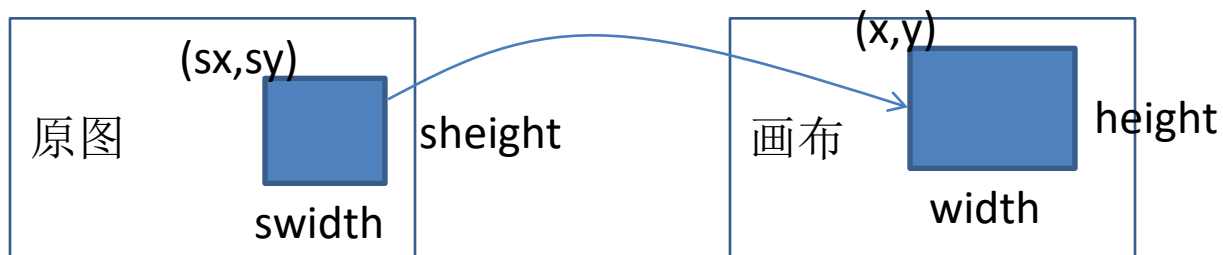
```
// 方法2、把原图复制到位置<x,y>处，图的宽度和高度缩放为width和height
```

```
context.drawImage(img,x,y,width,height);
```

```
// 方法3、把原图从<sx,sy>开始宽度为swidth和高度为sheight的区域复制到
```

```
// 位置<x,y>处，图的宽度和高度缩放为width和height
```

```
context.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);
```

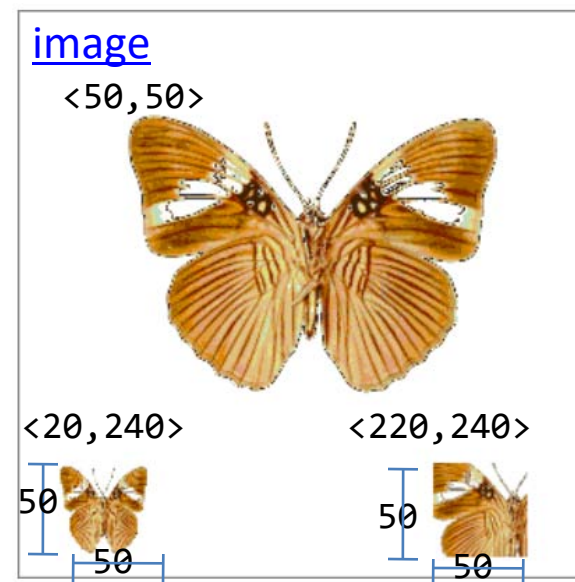


[参考](#)

```

<!DOCTYPE html>
<html>
<body>
  <canvas id="canvas" style="border:solid 1px gray;"
    height="400" width="400" >
    canvas is not supported!
  </canvas>
  <script>
    var can=document.getElementById("canvas");
    var context=canvas.getContext('2d');
    var img = new Image();
    img.addEventListener("load",
      function(){
        context.drawImage(img,50,50);           // 目标:x,y
        context.drawImage(img,20,240,50,50);    // 目标:x,y,width,height
        // 源: sx,sy,swidth,sheight 目标: x,y,width,height
        context.drawImage(img,20,20,100,100,220,240,50,50);
      },
      false);
    img.src="image/butterfly-1.gif";
  </script>
</body>
</html>

```



● 鼠标作画

```
<!DOCTYPE html><html>
<body> <h1>canvas鼠标画图</title>
    <canvas id="c1" width="400" height="400"
        style="border: solid 1px gray;">
    </canvas>
</body>
<script>
    var canvas1 = document.getElementById('c1');
    var can1 = canvas1.getContext('2d');
    canvas1.onmousedown = function (ev) {
        var ev = ev || window.event||arguments.callee.caller.arguments[0];
        can1.moveTo(ev.clientX - canvas1.offsetLeft, ev.clientY - canvas1.offsetTop+20);
        canvas1.onmousemove = function (ev) {
            var ev = ev || window.event||arguments.callee.caller.arguments[0];
            can1.lineTo(ev.clientX-canvas1.offsetLeft,ev.clientY-canvas1.offsetTop+20);
            can1.stroke();
        };
        canvas1.onmouseup = function () {
            canvas1.onmousemove = null;
            canvas1.onmouseup = null;
        };
    };
</script>
</html>
```



• 保存Canvas为文件

```
<!DOCTYPE html><html><body><h1>保存canvas</h1>
  <canvas id="c1" width="200" height="200"
    style="border: solid 1px gray;">
  </canvas>
  <p><input type="button" value="保存canvas为图片"
    onclick="saveImg(canvas1, 'save-canvas-to-image.png')"/>
  </p></body>
</html>

<script> var canvas1 = document.getElementById('c1');
  var ctx = canvas1.getContext('2d');
  ctx.strokeStyle="red";
  ctx.strokeRect(50,50,100,100);
  var saveImg = function (canvas, filename) {
    if (canvas.msToBlob) {
      var blob = canvas.msToBlob();
      window.navigator.msSaveBlob(blob, filename);
    } else {
      var save_link=document.createElementNS('http://www.w3.org/1999/xhtml','a');
      save_link.href=canvas.toDataURL('image/png');
      save_link.download = filename;
      var event = document.createEvent('MouseEvents');
      event.initMouseEvent('click', true, false, window, 0, 0, 0, 0, 0,
        false, false, false, false, 0, null);
      save_link.dispatchEvent(event);
    }
  };
</script></html>
```

[saveImage](#)



保存canvas为图片

//canvas保存为图片
//IE9+浏览器, Edge
//firefox,chrome

[参考123](#)

拖放操作

● 概述

[参考](#)

HTML5提供了拖放(drag and drop)支持。使用拖放功能，你可以把一个HTML元素拖放到另一个元素上。需要拖放的元素需要设置属性draggable为true。

```
<div class="myclass" draggable="true">Some content</div>
```

目标元素



drag事件发生时通过
`event.target`取得被拖对象。



drop事件发生时可以通过
`event.target`取得被放对象。注意始终防止被放对象的默认行为发生。

• 元素拖放

```
<!DOCTYPE HTML><html>
<head><meta char="utf-8">
<style> #div1{width:300px;height:80px;padding:8px;border:solid;}</style>
<script>
    function allowDrop(ev){ // dragover事件规定在何处放置被拖动数据
        ev.preventDefault(); // 只有阻止对元素的默认处理方式，才能放入其他元素
    }
    function drag(ev){
        ev.dataTransfer.setData("Text",ev.target.id);
    }
    function drop(ev){ // drop事件的默认动作是打开链接
        ev.preventDefault();
        var data=ev.dataTransfer.getData("Text");
        ev.target.appendChild(document.getElementById(data));
    }
</script>
</head>
<body>
    <p>请把蝴蝶图片拖放到矩形中:</p>
    <div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div><br>
    
</body>
</html>
```



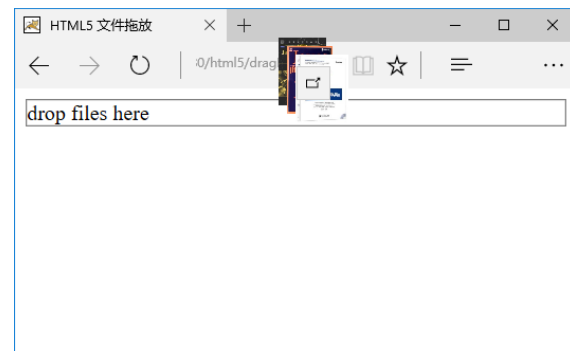
// 其他事件: ondropenter(ev), ondropleave(ev), ondropend(ev)

[参考](#)

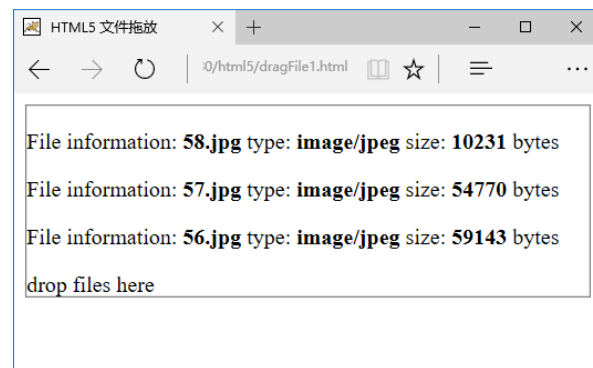
• 文件拖放

```
<!DOCTYPE html><html> <head><title>文件拖放</title>
<script type="text/javascript">
    function FileDragHover(e) {
        e.stopPropagation();
        e.preventDefault();
        e.target.className=(e.type=="dragover"?"hover":"","");
    }
    function FileSelectHandler(e) {
        FileDragHover(e);
        var files = e.target.files || e.dataTransfer.files;
        for (var i = 0, f; f = files[i]; i++) {
            ParseFile(f);
        }
    }
    function Output(msg) {
        var m = document.getElementById("filedrag");
        m.innerHTML = msg + m.innerHTML;
    }
    function ParseFile(file) {
        Output("<p>File information:" + file.name
            + " type:" + file.type + " size:" + file.size + " bytes</p>");
    }
</script></head>
<body> <div id="filedrag" ondragover="FileDragHover(event)"
        ondragleave="FileDragHover(event)" ondrop="FileSelectHandler(event)"
        style="border: 1px solid gray">drop files here</div>
</body></html>
```

dragFile



拖入三个文件后



Ajax文件上传

- 概述

(1) 把input控件中选择的多个文件加入到FormData对象:

```
var fd = new FormData();  
var files = document.getElementById('fileName').files;  
for (var i = 0, f; f = files[i]; i++) {  
    fd.append(f.name, f);  
}
```

(2) 建立一个Ajax对象，绑定load事件并把FormData对象提交给服务器:

```
var xhr = new XMLHttpRequest();  
xhr.addEventListener("load", uploadComplete, false);  
xhr.open("POST", "fileupload.jsp");  
xhr.send(fd);
```

(3) load事件发生时表示上传成功。还可以把xhr绑定error事件、progress事件、abort事件，表示出错、进度和放弃。

```

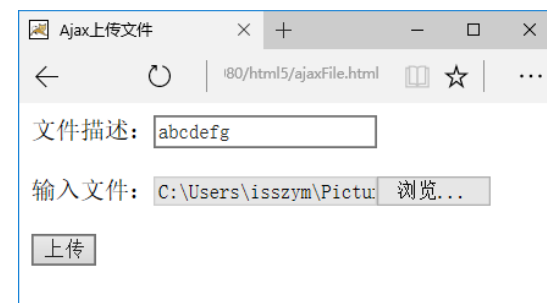
<!DOCTYPE html><html><head><title>Ajax上传文件</title>
<script type="text/javascript">
    function uploadFile() {
        var files = document.getElementById("myfile").files;
        var fd = new FormData();

        var desc=document.getElementById("desc");
        fd.append("desc", desc.value); // 增加表单数据

        for (var i = 0, f; f = files[i]; i++) {
            fd.append(f.name, f); // 增加文件
        }
        var xhr = new XMLHttpRequest();
        xhr.open("post", "fileUpload.jsp", true);
        xhr.onload = function () {
            alert("上传完成!" + event.target.responseText);
        };
        xhr.send(fd);
    }
</script></head>
<body>
    文件描述: <input type="text" id="desc" value=""/><br><br>
    输入文件: <input type="file" id="myfile" multiple /><br><br>
    <input type="button" onclick="uploadFile()" value="上传" />
</body>
</html>

```

[ajaxFile](#)



* 比较完整的例子见附录

fileUpload.jsp

```
<%@ page pageEncoding="utf-8" contentType="text/html; charset=utf-8"%>
<%@ page import="java.io.*"%> <%@ page import="java.util.*"%>
<%@ page import="org.apache.commons.io.*"%>
<%@ page import="org.apache.commons.fileupload.*"%>
<%@ page import="org.apache.commons.fileupload.disk.*"%>
<%@ page import="org.apache.commons.fileupload.servlet.*"%>
<html>
<head>
    <title>文件传输例子</title>
</head>
<body>
<%request.setCharacterEncoding("utf-8");%>
<% boolean isMultipart
    = ServletFileUpload.isMultipartContent(request); //检查表单中是否包含文件
if (isMultipart) {
    FileItemFactory factory = new DiskFileItemFactory();
    //factory.setSizeThreshold(yourMaxMemorySize); //设置使用的内存最大值
    //factory.setRepository(yourTempDirectory); //设置文件临时目录
    ServletFileUpload upload = new ServletFileUpload(factory);
    //upload.setSizeMax(yourMaxRequestSize); //允许的最大文件尺寸
    List items = upload.parseRequest(request);
```

```

for (int i = 0; i < items.size(); i++) {
    FileItem fi = (FileItem) items.get(i);
    if (fi.isFormField()) { //如果是表单字段
        out.println("输入域:"+fi.getFieldName()+" 取值:"+fi.getString("utf-8"));
    }
    else { //如果是文件
        DiskFileItem dfi = (DiskFileItem) fi;
        if (!dfi.getName().trim().equals("")) { //getName()返回文件名称或空串
            out.print("文件被上传到服务上的实际位置: ");
            String fileName = application.getRealPath("/file")
                                + System.getProperty("file.separator")
                                + FilenameUtils.getName(dfi.getName());
            out.println(new File(fileName).getAbsolutePath());
            dfi.write(new File(fileName));
        }
    }
} //for
} //if
%></body></html>

```

HTML5提供的XMLHttpRequest Level 2已经实现了跨域访问以及其他的一些新功能，这需要在远程服务器端添加如下代码：

```

header('Access-Control-Allow-Origin:'); /*代表可访问的地址，可以设置指定域名
header('Access-Control-Allow-Methods:POST,GET');

```

[拖放上传](#) [拖放上传2](#)

IndexedDB

Web Storage（Local Storage和Session Storage）可以用于本地存储一些键值对，方便灵活，但是对于大量结构化数据存储还是力不从心。IndexedDB可以用于客户端存储大量的结构化数据，并且可以使用索引实现快速检索。IndexedDB采用请求响应模式实现数据库操作，并用js对象进行保存数据。

[参考1](#) [2](#) [3](#) [4](#)

打开或创建
数据库

```
openRequest = indexedDB.open("persons",1); { //库名,版本号  
db = event.target.result;
```

获得事务

```
var transaction =  
db.transaction(["person"], "readwrite");
```

获得对象仓库

```
var objectStore =  
transaction.objectStore("person");
```

处理数据行

```
objectStore.openCursor(); //取得数据  
objectStore.add(person); //加入数据（person对象）  
objectStore.get(removeKey); //删除数据（按键值）
```

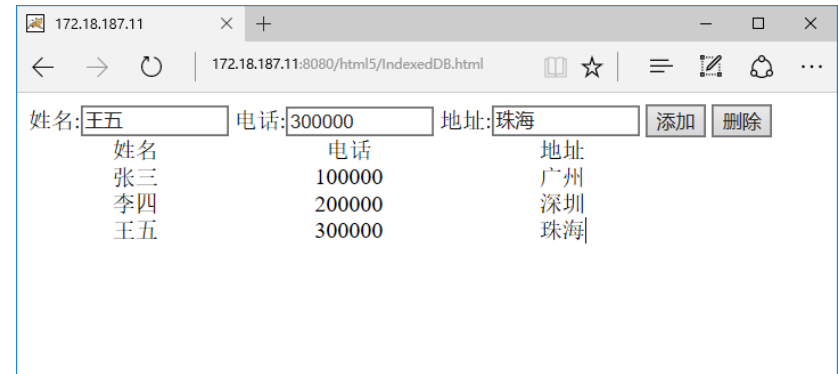
```

<!DOCTYPE html>
<html>
  <head>
    <script>
      .....
    </script>
    <style type="text/css">
      input {width:100px}
      div#content div {width:460px}
      div#head div, div#content div div {
        width: 150px;
        text-align: center;
        float: left;
      }
    </style>
  </head>
  <body>
    <label for="name">姓名:</label><input id="name" type="text" />
    <label for="phone">电话:</label><input id="phone" type="text" />
    <label for="address">地址:</label><input id="address" type="text" />
    <button id="add" accesskey="Enter">添加</button>
    <button id="delete">删除</button>

    <div id="head"><div>姓名</div> <div>电话</div><div>地址</div></div><br />
    <div id="content"></div>
  </body>
</html>

```

IndexedDB.html



indexedDB通过版本号的变化表示要创建或升级数据库。**indexedDB**数据库中保存的是js对象。**keyPath**指出主键值，该主键不必出现在js对象中。如果需要用某个数据域查询这些对象，要用**createIndex**为该数据域创建索引。

```
// https://developer.mozilla.org/en/IndexedDB/Using\_IndexedDB
var db, arrayKey=[], openRequest, lastCursor;
var indexedDB = window.indexedDB || window.webkitIndexedDB
                || window.mozIndexedDB || window.msIndexedDB;
function init(){
    openRequest = indexedDB.open("persons",1);           //数据库名,版本号
    openRequest.onupgradeneeded = function(e) {         //每次修改版本号执行
        var thisDb = e.target.result;
        if(!thisDb.objectStoreNames.contains("person")) {
            var objectStore = thisDb.createObjectStore("person",
                {keyPath: "id", autoIncrement:true});
            objectStore.createIndex("name", "name", { unique: false });
        }
    }
}
```

```

// 当打开(数据库或游标)请求成功时
openRequest.onsuccess = function(e) {
    db = e.target.result;
    if(db.objectStoreNames.contains("person")){
        var transaction = db.transaction(["person"],"readwrite");
        var content= document.querySelector("#content");
        var objectStore = transaction.objectStore("person");
        var request = objectStore.openCursor();
        request.onsuccess = function(event){
            var cursor = event.target.result;
            var flag=0;
            if (cursor){
                addRow(content,cursor.key,cursor.value[flag]["name"],
                    cursor.value[flag]["phone"],cursor.value[flag]["address"]);
                arrayKey.push(cursor.key);
                flag++;
                lastCursor=cursor.key;
                cursor.continue(); // 该游标移动到下一个对象
            } //if
        };
    } //if
}; //openRequest

```

```
var request = ObjectStore.openCursor(query, direction);
```

参数: query: 可选。单个键值或所有记录的键值（默认）

direction: 可选。。取值: "next"（默认）,"nextunique","prev","prevunique"

返回: 一个激活关联操作的后续事件的IDBRequest对象。


```

//add new record
document.querySelector("#add").addEventListener("click", function() {
    var name=document.querySelector("#name").value;
    var phone=document.querySelector("#phone").value;
    var address=document.querySelector("#address").value;
    var person=[{"name":name,"phone":phone,"address":address}]
    var transaction = db.transaction(["person"], "readwrite");
    var objectStore = transaction.objectStore("person");
    objectStore.add(person);
    var request = objectStore.openCursor();
    request.onsuccess = function(event) {
        cursor = event.target.result;
        var key;
        if(lastCursor==null) {
            key=cursor.key;
            lastCursor=key;
        }
        else {
            key=++lastCursor;
        }
        addRow(content,key,name,phone,address);
        arrayKey.push(key);
    }
});

```

```

document.querySelector("#delete").addEventListener("click", function() {
    if(arrayKey.length==0){ console.log("no data to delete!"); }
    else {
        var transaction = db.transaction(["person"], "readwrite");
        var objectStore = transaction.objectStore("person");
        var removeKey=arrayKey.shift();
        var getRequest=objectStore.get(removeKey);
        getRequest.onsuccess=function(e) {
            var request=objectStore.delete(removeKey);
            document.getElementById(removeKey).style.display="none";
        }
    }
});
} //init
function addRow(conntent,key,name,phone,address){
    var div=document.createElement("div");
    div.id=key;
    var div1=document.createElement("div");
    var div2=document.createElement("div");
    var div3=document.createElement("div");
    div1.innerHTML=name; div2.innerHTML=phone;div3.innerHTML=address;
    div.appendChild(div1);div.appendChild(div2);div.appendChild(div3);
    content.appendChild(div);
};
window.addEventListener("DOMContentLoaded", init, false);

```

文件API

采用input元素选中文件后，可以通过文件API直接在网页上把它们显示出来。

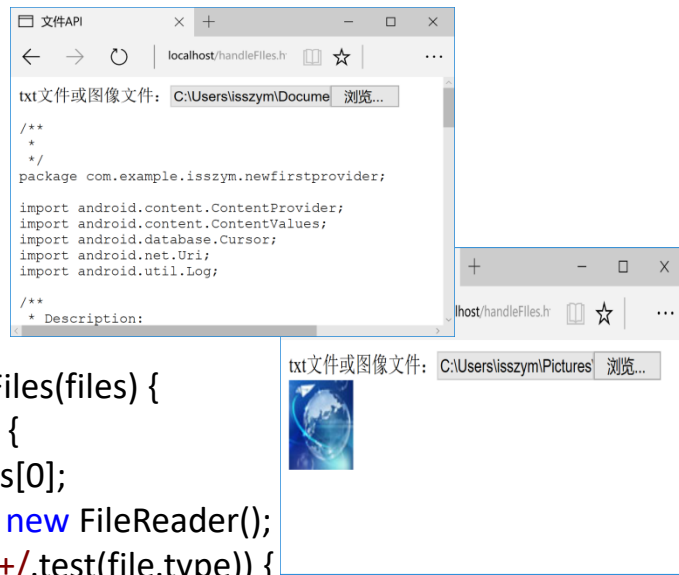
```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type"
    content="text/html; charset=utf-8" />
  <title>文件API</title>
  <style>
    body { background-color: yellow; }
    #c1 { background-color: white; }
  </style>
</head>
<body>
  txt文件或图像文件:
  <input type="file"
    id="file" onchange="handleFiles(this.files)"/>
  <div id="content"></div>
</body>
</html>
```

fileAPI

<script>

```
function handleFiles(files) {
  if (files.length) {
    var file = files[0];
    var reader = new FileReader();
    if (/text\w+/.test(file.type)) {
      reader.onload = function () {
        var div = document.getElementById("content");
        div.innerHTML = '<pre>' + this.result + '</pre>';
      }
      reader.readAsText(file);
    } else if (/image\w+/.test(file.type)) {
      reader.onload = function () {
        var div = document.getElementById("content");
        div.innerHTML = '';
      }
      reader.readAsDataURL(file);
    }
  }
}
```

</script>



[参考](#)

离线存储

- html5可以通过离线存储功能把网页对象存储在本地，使得主机在没有连网时也可以正常浏览这些网页，结合本地存储就可以完成很多功能。
- 在html标签里通过manifest属性引用一个cache.manifest文件，该文件里声明了浏览器需缓存的所有文件。

```
<!DOCTYPE html>
<html lang="en" manifest="demo.appcache">
<head>
  <meta charset="UTF-8">
  <title>HTML5离线存储</title>
</head>
<body>
  
  
</body>
</html>
```

CACHE MANIFEST

#v01

image/01.jpg

NETWORK:

*

FALLBACK:

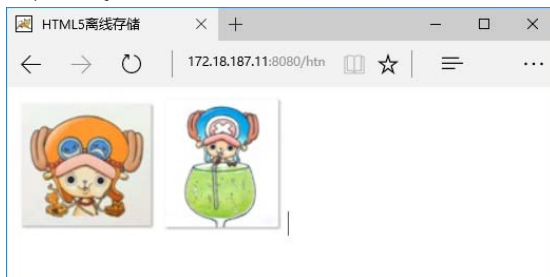
/

列出离线要保存的文件
(除了html文件)

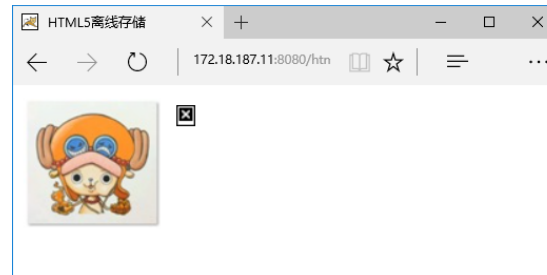
在线时要下载的文件
(除了前面的文件)

URL无法访问时的替代页面

在线



离线时只显示一幅图像 (01.jpg)



[参考1](#) [2](#) [3](#) [4](#) [5](#)

- manifest文件是简单的文本文件，分为三个部分：
 - CACHE MANIFEST - 列出URL的文件将在首次下载后进行缓存。
 - NETWORK - 列出URL的文件需要与服务器的连接，且不会使用本地存储。
 - FALLBACK - 给出在指定URL无法访问时的替代页面。其中，#为注释，*表示除了以前已定义文件之外的所有文件。
- 在线的情况下，用户代理每次访问页面，都会去读一次manifest。如果发现其改变, 则重新加载全部清单中的资源。

CACHE MANIFEST

/main.css
/logo.jpg
/main.js

NETWORK:

/login.jsp

FALLBACK:

/html5/ /404.html

• 离线存储的注意事项

- 站点离线存储的容量限制是5M
- 如果manifest文件，或者内部列举的某一个文件不能正常下载，整个更新过程将视为失败，浏览器继续全部使用老的缓存。
- 引用manifest的html必须与manifest文件同源，在同一个域名下。
- 在manifest中使用的相对路径，相对参照物为manifest文件。
- CACHE MANIFEST字符串应在第一行，且必不可少。
- 系统会自动缓存引用清单文件的 HTML 文件。
- manifest文件中CACHE则与NETWORK，FALLBACK的位置顺序没有关系，如果是隐式声明需要在最前面
- FALLBACK中的资源必须和manifest文件同源
- 当一个资源被缓存后，该浏览器直接请求这个绝对路径也会访问缓存中的资源。
- 站点中的其他页面即使没有设置manifest属性，请求的资源如果在缓存中也从缓存中访问。
- 当manifest文件发生改变时，资源请求本身也会触发更新。

WebSocket



建立连接:

```
socket=new WebSocket(host)
```

```
host { ws://host/html5/webs (http)
      wss://host/html5/webs (https) }
```

```
事件 { socket.onmessage 数据到达
      socket.onclose   连接关闭 }
```

```
方法  socket.send(message)
```

虚目录html5下的webs包中的一个.class文件包含下面语句:

```
@ServerEndpoint(value = "/webs")
```

每个连接得到该类的一个包含连接会话的线程实例:

```
事件 { @OnOpen    连接建立
      @OnClose  连接关闭
      @OnMessage 数据到达 }
```

```
方法 { sendText
      sendBinary } 发送给连接会话
      sendObject 的远端对象
      close      连接会话
```

客户端程序(chat1.html):

[chat1](#)

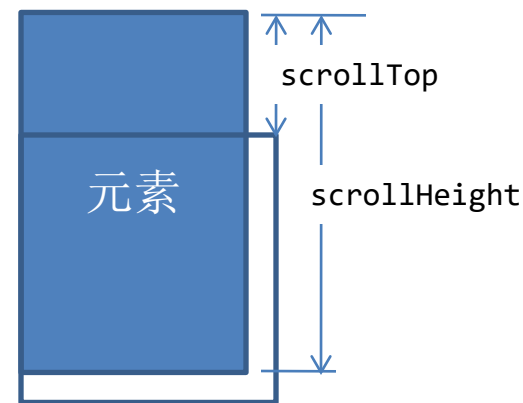
```
<!DOCTYPE html><html>
  <head><title>Apache Tomcat WebSocket Examples: Chat1</title>
    <style type="text/css">
      input#chat { width: 410px }
      #console-container { width: 400px; }
      #console {
        border: 1px solid #CCCCCC;
        border-right-color: #999999;
        border-bottom-color: #999999;
        height: 170px;
        overflow-y: scroll;
        padding: 5px;
        width: 100%;
      }
      #console p { padding: 0; margin: 0; }
    </style>
  </head>
  <body>
    <p><input type="text" placeholder="type and press enter to chat" id="chat"/></p>
    <div id="console-container">
      <div id="console"/></div>
    </div>
    <script type="application/javascript">
      //见下页
    </script>
  </body>
</html>
```



```

var host = 'ws://' + window.location.host + '/html5/webs'; //ws:(http)=>wss:(https)
var socket = (new WebSocket(host))||(new MozWebSocket(host));
if(socket==null) {log("Info: WebSocket Error.");}
else {
    log("Info: WebSocket opened.");
    document.getElementById('chat').onkeydown = function(event) {
        if (event.keyCode == 13) {sendMessage();}
    };
    socket.onclose = function () { log('Info: WebSocket closed. ');
        document.getElementById('chat').onkeydown = null;
    };
    socket.onmessage = function (message) {log(message.data); };
    function sendMessage(){
        var message = document.getElementById('chat').value;
        if (message != '') {
            socket.send(message);
            document.getElementById('chat').value = '';
        }
    };
}
function log(message) {
    var console = document.getElementById('console');
    var p = document.createElement('p');
    p.style.wordWrap = 'break-word'; p.innerHTML = message;
    console.appendChild(p);
    while (console.childNodes.length > 25) {
        console.removeChild(console.firstChild);
    }
    console.scrollTop = console.scrollHeight; // 尽可能上滚
};

```



浏览器客户区

* 元素高度小于客户区高度不会上滚，大于客户区高度而上滚时不会让底部出现空白区。

服务器程序/WEB-INF/classes/webs/ChatExample.class

```
package webs;
import java.io.IOException; import java.util.Set;
import java.util.concurrent.CopyOnWriteArraySet;
import java.util.concurrent.atomic.AtomicInteger;
import javax.websocket.OnClose;import javax.websocket.OnError;
import javax.websocket.OnMessage; import javax.websocket.OnOpen;
import javax.websocket.Session; import javax.websocket.server.ServerEndpoint;
import org.apache.juli.logging.Log; import org.apache.juli.logging.LogFactory;
```

```
@ServerEndpoint(value = "/webs")
```

```
public class ChatExample {
    private static final Log log = LogFactory.getLog(ChatExample.class);
    private static final String GUEST_PREFIX = "Guest";
    private static final AtomicInteger connectionIds = new AtomicInteger(0);
    private static final Set<ChatExample> connections
        = new CopyOnWriteArraySet<>();

    private final String nickname;
    private Session session;
    public ChatExample() {
        nickname = GUEST_PREFIX + connectionIds.getAndIncrement();
    }
}
```

```

@OnOpen
public void start(Session session) {
    this.session = session;
    connections.add(this);
    String message = String.format("* %s %s", nickname, "has joined.");
    broadcast(message);
}

@OnClose
public void end() {
    connections.remove(this);
    String message = String.format("* %s %s",
        nickname, "has disconnected.");
    broadcast(message);
}

@OnMessage
public void incoming(String message) {
    // Never trust the client
    String filteredMessage = String.format("%s: %s",
        nickname, filter(message.toString()));
    broadcast(filteredMessage);
}

```

```

@OnError
public void onError(Throwable t) throws Throwable {
    Log.error("Chat Error: " + t.toString(), t);
}

private static void broadcast(String msg) {
    for (ChatExample client : connections) {
        try {
            synchronized (client) { // 对象同步
                client.session.getBasicRemote().sendText(msg);
            }
        } catch (IOException e) {
            Log.debug("Chat Error: Failed to send message to client", e);
            connections.remove(client);
            try {
                client.session.close();
            } catch (IOException e1) {
                // Ignore
            }
            String message = String.format("* %s %s",
                client.nickname, "has been disconnected.");
            broadcast(message);
        }
    }
}
}
}

```

[WebSocket session参考](#)

编译命令参考: C:>javac -classpath ./websocket-api.jar;./tomcat-coyte.jar;./tomcat-juli.jar ChatExample.java
 * jar包可以在tomcat下找到

Geolocation API

Geolocation API用于将用户当前地理位置信息共享给信任的站点，这涉及用户的隐私安全问题，所以当站点需要获取用户的当前地理位置，浏览器会提示用户是“允许” or “拒绝”。

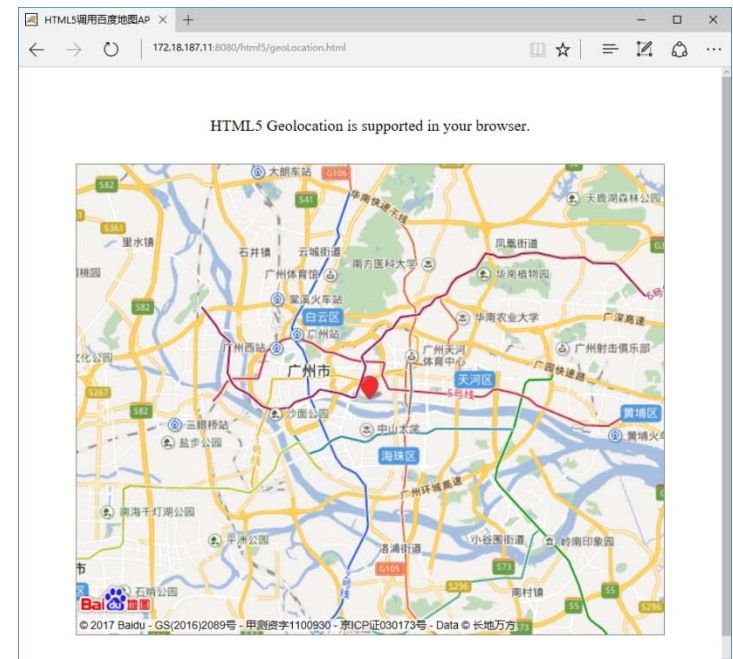
先看看哪些浏览器支持Geolocation API: IE9.0+、FF3.5+、Safari5.0+、Chrome5.0+、Opera10.6+、iPhone3.0+、Android2.0+

```
<!DOCTYPE html> <html> <title>HTML5调用百度地图API进行地理定位实例</title> <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <script type="text/javascript"
    src="http://api.map.baidu.com/api?v=2.0&ak=134db1b9cf1f1f2b4427210932b34dcb">
  </script>
</head>
<body style="margin:50px 10px;">
  <div id="status" style="text-align: center"></div>
  <div style="width:600px;height:480px;border:1px solid gray;margin:30px auto"
    id="container"></div>
</body>
</html>
```

```

<script type="text/javascript">
    var x=121.48789949,y=31.24916171; //默认地理位置设置为上海市浦东新区
    window.onload = function() {
        if(navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition);
            document.getElementById("status").innerHTML = "Geolocation is supported.";
            var map = new BMap.Map("container"); // 百度地图API功能
            var point = new BMap.Point(x,y);
            map.centerAndZoom(point,12);           // 中心点，放大倍数
            var geolocation = new BMap.Geolocation();
            geolocation.getCurrentPosition(function(r){
                if(this.getStatus() == BMAP_STATUS_SUCCESS){
                    var mk = new BMap.Marker(r.point); // 打标记
                    map.addOverlay(mk);
                    map.panTo(r.point);
                }
                else {
                    alert('failed'+this.getStatus());
                }
            },{enableHighAccuracy: true})
            return;
        }
        alert("你的浏览器不支持获取地理位置！");
    };
    function showPosition(position){
        x=position.coords.latitude;
        y=position.coords.longitude;
    }
</script>

```



HTML5的其它功能

PhoneGap

PhoneGap是一个用基于HTML, CSS和JavaScript的, 创建移动跨平台移动应用程序的快速开发平台。它使开发者能够利用IOS, Android, Palm, Symbian, WP7, WP8, Bada和Blackberry智能手机的核心功能——包括地理定位, 加速器, 联系人, 声音和振动等, 此外PhoneGap拥有丰富的插件可以调用。

业界很多主流的移动开发框架均源于PhoneGap。较著名的有Worklight、appMobi、WeX5、H5等; 其中H5和WeX5为国内打造, 完全Apache开源, 在融合Phonegap的基础上, 做了深度优化, 具备接近Native app的性能, 同时开发便捷性也较好。

WebRTC

[WebRTC](#)，名称源自网页实时通信（Web Real-Time Communication）的缩写，是一个支持网页浏览器进行实时语音对话或视频对话的技术，是谷歌2010年以6820万美元收购Global IP Solutions公司而获得的一项技术。2011年5月开放了工程的源代码，在行业内得到了广泛的支持和应用，成为下一代视频通话的标准。

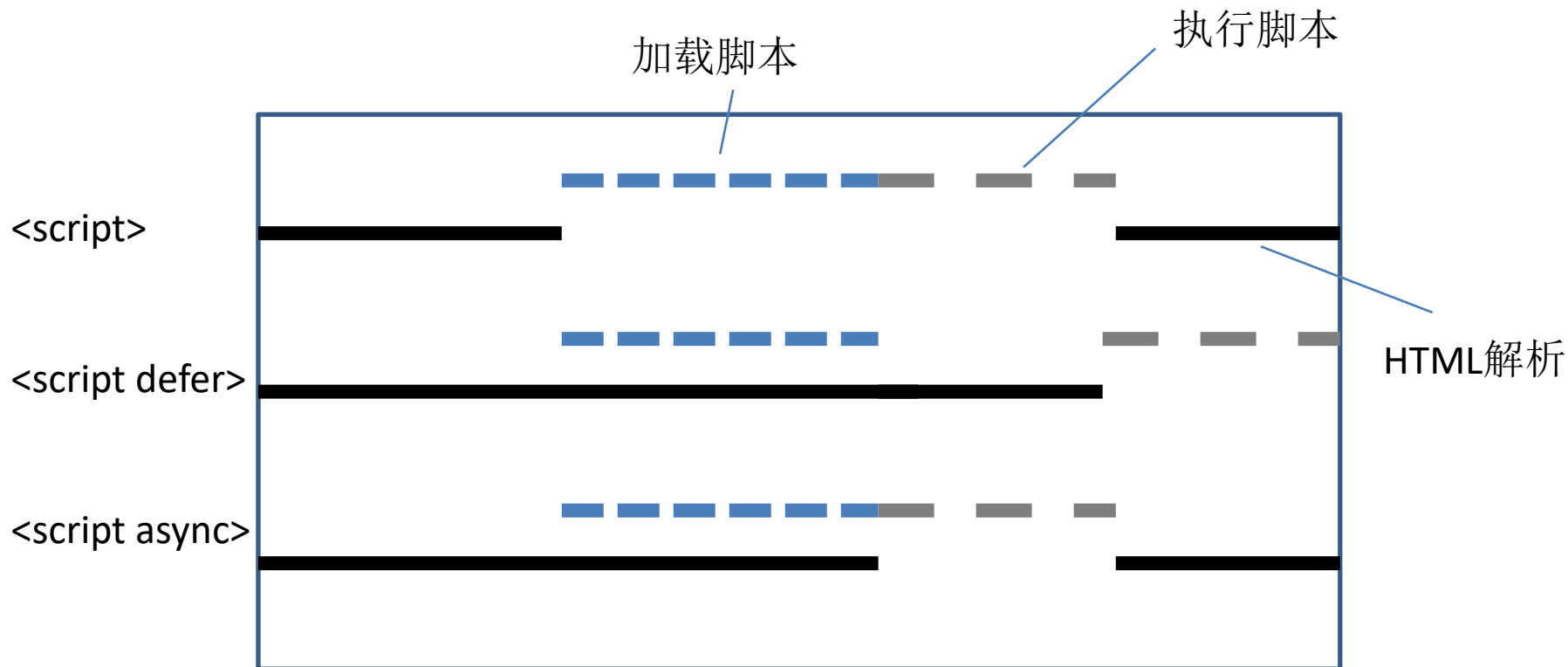
SVG

SVG（Scalable Vector Graphics，可缩放矢量图形）是基于可扩展标记语言（标准通用标记语言的子集），用于描述二维矢量图形的一种图形格式。它由万维网联盟制定，是一个开放标准。Internet Explorer9，火狐，谷歌Chrome，Opera和Safari都支持SVG。

延迟执行JavaScript文件

- 在默认情况下，浏览器顺序加载和解析所有页面元素，这会减慢HTML的解析。HTML5为script元素增加了属性defer和async来克服这个问题。另一种解决方法是把script元素放在页面的末尾（</body>之前）。

`<script src="main.js" defer> </script>`



附录1、 canvas

颜色、样式和阴影

属性	描述
fillStyle	设置或返回用于填充绘画的颜色、渐变或模式
strokeStyle	设置或返回用于笔触的颜色、渐变或模式
shadowColor	设置或返回用于阴影的颜色
shadowBlur	设置或返回用于阴影的模糊级别
shadowOffsetX	设置或返回阴影距形状的水平距离
shadowOffsetY	设置或返回阴影距形状的垂直距离

方法	描述
createLinearGradient()	创建线性渐变（用在画布内容上）
createPattern()	在指定的方向上重复指定的元素
createRadialGradient()	创建放射状/环形的渐变（用在画布内容上）
addColorStop()	规定渐变对象中的颜色和停止位置

线条样式

属性	描述
lineCap	设置或返回线条的结束端点样式
lineJoin	设置或返回两条线相交时，所创建的拐角类型
lineWidth	设置或返回当前的线条宽度
miterLimit	设置或返回最大斜接长度

矩形

方法	描述
rect()	创建矩形
fillRect()	绘制“被填充”的矩形
strokeRect()	绘制矩形（无填充）
clearRect()	在给定的矩形内清除指定的像素

路径

方法	描述
<u>fill()</u>	填充当前绘图（路径）
<u>stroke()</u>	绘制已定义的路径
<u>beginPath()</u>	起始一条路径，或重置当前路径
<u>moveTo()</u>	把路径移动到画布中的指定点，不创建线条
<u>closePath()</u>	创建从当前点回到起始点的路径
<u>lineTo()</u>	添加一个新点，然后在画布中创建从该点到最后指定点的线条
<u>clip()</u>	从原始画布剪切任意形状和尺寸的区域
<u>quadraticCurveTo()</u>	创建二次贝塞尔曲线
<u>bezierCurveTo()</u>	创建三次方贝塞尔曲线
<u>arc()</u>	创建弧/曲线（用于创建圆形或部分圆）
<u>arcTo()</u>	创建两切线之间的弧/曲线
<u>isPointInPath()</u>	如果指定的点位于当前路径中，则返回 true ，否则返回 false

转换

方法	描述
scale()	缩放当前绘图至更大或更小
rotate()	旋转当前绘图
translate()	重新映射画布上的 (0,0) 位置
transform()	替换绘图的当前转换矩阵
setTransform()	将当前转换重置为单位矩阵。然后运行 transform()

文本

属性	描述
font	设置或返回文本内容的当前字体属性
textAlign	设置或返回文本内容的当前对齐方式
textBaseline	设置或返回在绘制文本时使用的当前文本基线

方法	描述
fillText()	在画布上绘制“被填充的”文本
strokeText()	在画布上绘制文本（无填充）
measureText()	返回包含指定文本宽度的对象

图像绘制

方法	描述
drawImage()	向画布上绘制图像、画布或视频

像素操作

属性	描述
width	返回 ImageData 对象的宽度
height	返回 ImageData 对象的高度
data	返回一个对象，其包含指定的 ImageData 对象的图像数据

方法	描述
createImageData()	创建新的、空白的 ImageData 对象
getImageData()	返回 ImageData 对象，该对象为画布上指定的矩形复制像素数据
putImageData()	把图像数据（从指定的 ImageData 对象）放回画布上

合成

属性	描述
globalAlpha	设置或返回绘图的当前 alpha 或透明值
globalCompositeOperation	设置或返回新图像如何绘制到已有的图像上

其他

方法	描述
<code>save()</code>	保存当前环境的状态
<code>restore()</code>	返回之前保存过的路径状态和属性
<code>createEvent()</code>	
<code>getContext()</code>	
<code>toDataURL()</code>	

附录2、拖放操作

拖放操作的事件：

事件	描述
dragstart	在拖(drag)操作开始时发生。
drag	当一个元素被拖时该事件发生。
dragenter	当一个元素被拖入一个可放入的目标元素(valid drop target)时发生。
dragleave	当一个元素被拖离一个可放入的目标元素时发生。
dragover	当一个元素被拖到一个可放入的目标元素上方时发生。
drop	当一个元素被放到一个可放入的目标元素时发生。
dragend	在拖(drag)操作结束时发生。

```
var src=document.getElementById("drag1");
src.addEventListener('dragstart', drag(event), false);

var dest=document.getElementById("drop1");
dest.addEventListener('drop', drop(event), false);
dest.addEventListener('dragover', allowDrop(event), false);
```

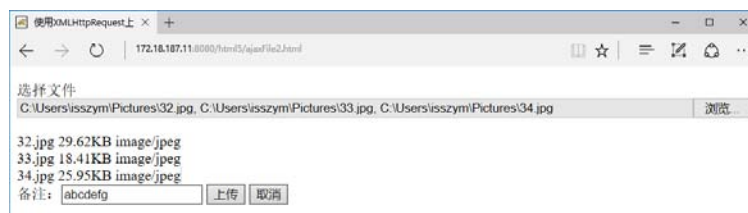

拖放操作的属性/方法:

属性 / 方法	描述
effectAllowed	指出允许的操作类型。 可能的取值: none, copy, copyLink, copyMove, link, linkMove, move, all, uninitialized。
dropEffect	指出当前选择的操作类型(需要effectAllowed支持)。 可能的取值: none, copy, link, move。
setDragImage()	设置拖动时的鼠标图片。
setData()	设置拖动操作要传送的数据。
getData()	取到拖动操作传送过来的数据。
clearData()	删除前面存储的拖动数据。

附录3、Ajax多文件上传完整例子

```
<%@page language="java" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<% String path = request.getContextPath();%>
<!DOCTYPE html><html><head><title>使用XMLHttpRequest上传文件</title>
<script type="text/javascript">
    var xhr = new XMLHttpRequest();
    function fileSelected() {
        var msg = "";
        var files = document.getElementById('fileName').files;
        for (var i = 0, file; file = files[i]; i++) {
            var fileSize = 0;
            if (file.size > 1024 * 1024)
                fileSize = (Math.round(file.size*100/(1024*1024))/100).toString()+ 'MB';
            else
                fileSize = (Math.round(file.size*100/1024)/100).toString()+ 'KB';
            msg = msg + file.name
                + ' ' + fileSize + ' ' + file.type + "<br>";
        }
        document.getElementById('msg').innerHTML = msg;
    }
}
```

[ajaxFile2](#)



```

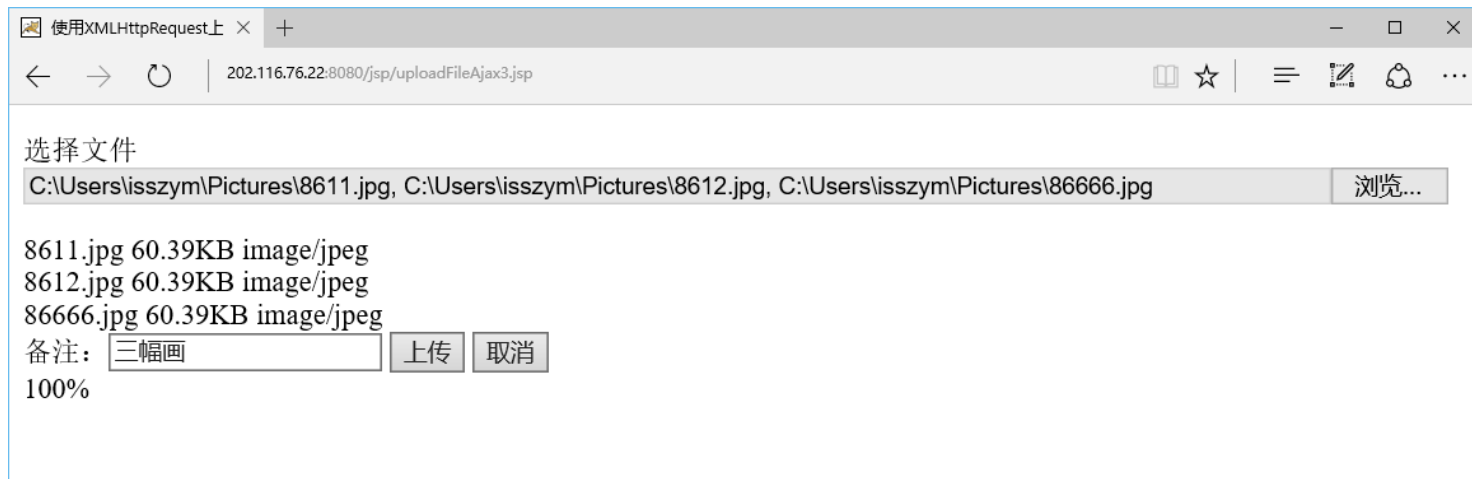
function uploadFile() {
    var fd = new FormData();
    fd.append("memo", document.getElementById('memo').value);    //加入数据域memo
    var files = document.getElementById('fileName').files;
    for (var i = 0, f; f = files[i]; i++) {
        fd.append(f.name, f);
    }
    xhr.upload.addEventListener("progress", uploadProgress, false);
    xhr.addEventListener("load", uploadComplete, false);
    xhr.addEventListener("error", uploadFailed, false);
    xhr.addEventListener("abort", uploadCanceled, false);
    xhr.open("POST", "fileupload.jsp");
    xhr.send(fd);
}
function cancelUploadFile() { xhr.abort();}
function uploadProgress(evt) {
    if (evt.lengthComputable) {
        var percentComplete = Math.round(evt.loaded * 100 / evt.total);
        document.getElementById('progressNumber').innerHTML = percentComplete
            .toString()
            + '%';
    } else {
        document.getElementById('progressNumber').innerHTML = 'unable to compute';
    }
}
function uploadComplete(evt) {alert(evt.target.responseText);}
function uploadFailed(evt) {alert("上传失败"); }
function uploadCanceled(evt) {alert("您取消了本次上传.");}
</script>
</head>

```

```

<body>
  <p><label for="fileToUpload">选择文件</label> <input type="file"
    name="fileName" id="fileName" multiple onchange="fileSelected();"
    style="width: 800px" /></p>
  <div id="msg"></div>
  备注: <input type="text" name="memo" id="memo" />
  <input type="button" onclick="uploadFile()" value="上传" />
  <input type="button" onclick="cancleUploadFile()" value="取消" />
  <div id="progressNumber"></div>
</body>
</html>

```




先用“浏览”选择文件，再点击“上传”按钮

附录4、indexedDB完整源码

```
<!DOCTYPE html><html><head>
<script>
.....
</script>
<style type="text/css">
div#head div {
    width: 150px;
    padding: 0;
    margin: 1px;
    background-color: Yellow;
    text-align: center;
    font-family: @ 微软雅黑;
    font-size: larger;
    float: left;
}
#content {
    width: 460px;
}
#content div {
    width: 460px;
    text-align: center;
    background-color: Aqua;
    margin: 1px;
    float: left;
}
}
```

[indexedDB2](#)



姓名	电话	地址
aaa	12345678	东风路13号
李四	1366123456	大学城13号

```

#content div div {
    width: 150px;
    text-align: center;
    background-color: Aqua;
    margin: 1px;
    float: left;
}
</style>
</head>
<body> <span>姓名: </span>
    <input id="name" type="text" maxlength="5" />
    <span>电话: </span>
    <input id="phone" type="text" maxlength="12" />
    <span>地址: </span>
    <input id="address" type="text" maxlength="30" />
    <button id="add" accesskey="Enter">添加</button>
    <button id="delete">删除</button>
    <!--<button id="deleteDB">删除数据库</button>-->
    <div id="head">
        <div>姓名</div>
        <div>电话</div>
        <div>地址</div>
    </div>
    <br />
    <div id="content"></div>
</body>
</html>

```

```

<script>
  // https://developer.mozilla.org/en/IndexedDB/Using_IndexedDB
  var db;
  var arrayKey=[]
  var openRequest;
  var lastCursor;
  var indexedDB = window.indexedDB || window.webkitIndexedDB || window.mozIndexedDB
                  || window.msIndexedDB;
  function init() {
    openRequest = indexedDB.open("persons"); //handle setup
    openRequest.onupgradeneeded = function(e) {
      console.log("running onupgradeneeded");
      var thisDb = e.target.result;
      if(!thisDb.objectStoreNames.contains("person")) {
        console.log("I need to create the objectstore");
        var objectStore = thisDb.createObjectStore("person", { keyPath:
                                                                "id", autoIncrement:true });
        objectStore.createIndex("name", "name", { unique: false });
      }
    }
  }

```

```

openRequest.onsuccess = function(e) {
    db = e.target.result;
    db.onerror = function(event) {
        // Generic error handler for all errors targeted at this database's
        alert("Database error: " + event.target.errorCode);
        console.dir(event.target);
    };
    if(db.objectStoreNames.contains("person")){
        console.log("contains person");
        var transaction = db.transaction(["person"], "readwrite");
        transaction.oncomplete=function(event) {console.log("All done!");};
        var content= document.querySelector("#content");
        transaction.onerror = function(event){
            // Don't forget to handle errors!
            console.dir(event);
        };
        var objectStore = transaction.objectStore("person");
        //得到person的objectStore对象
    }
}

```



```
//得到person的objectStore对象
objectStore.openCursor().onsuccess = function(event){
    var cursor = event.target.result;
    var flag=0;
    if (cursor) {
        console.log(cursor.key);
        console.dir(cursor.value);
        var div=document.createElement("div");
        div.id=cursor.key;
        var div1=document.createElement("div");
        var div2=document.createElement("div");
        var div3=document.createElement("div");
        div1.innerHTML=cursor.value[flag]["name"];
        div2.innerHTML=cursor.value[flag]["phone"];
        div3.innerHTML=cursor.value[flag]["address"];
        div.appendChild(div1);
        div.appendChild(div2);
        div.appendChild(div3);
        content.appendChild(div);
        arrayKey.push(cursor.key);
        flag++;
        lastCursor=cursor.key;
        cursor.continue();
    } else {
        console.log("Done with cursor");
    }
};
```

```

openRequest.onerror = function (event) {
    // 对request.error做一些需要的处理!
    console.log("your web may not support IndexedDB,please check.");
};
//add new record
document.querySelector("#add").addEventListener("click", function() {
    var name=document.querySelector("#name").value;
    var phone=document.querySelector("#phone").value;
    var address=document.querySelector("#address").value;
    var person=[{"name":name,"phone":phone,"address":address}]
    var transaction = db.transaction(["person"], "readwrite");
    transaction.oncomplete = function(event) {
        // console.log("transaction complete");
    };
    transaction.onerror = function(event){
        console.dir(event);
    };
    var objectStore = transaction.objectStore("person");
    //得到person的objectStore对象
    objectStore.add(person);
    objectStore.openCursor().onsuccess = function(event) {
        cursor = event.target.result;
        var key;
        if(lastCursor==null){
            key=cursor.key;
            lastCursor=key;
        }
        else {
            key=++lastCursor;
        }
    }
}

```

```

        var content= document.querySelector("#content");
        var div=document.createElement("div");
        div.id=key
        var div1=document.createElement("div");
        var div2=document.createElement("div");
        var div3=document.createElement("div");
        div1.innerHTML=name;
        div2.innerHTML=phone;
        div3.innerHTML=address;
        div.appendChild(div1);
        div.appendChild(div2);
        div.appendChild(div3);
        content.appendChild(div);
        arrayKey.push(key);
        console.log("success add new record!key:"+key);
        console.dir(person);
    }
});

document.querySelector("#delete").addEventListener("click", function(){
    if(arrayKey.length==0){ console.log("no data to delete!"); }
    else{
        var transaction = db.transaction(["person"], "readwrite");
        transaction.oncomplete = function(event) {
            //console.log("transaction complete!");
        };
        transaction.onerror = function(event) {
            console.dir(event);
        };
    }
});

```

```

    var objectStore = transaction.objectStore("person");
                                //得到person的objectStore对象
    var removeKey=arrayKey.shift();
    var getRequest=objectStore.get(removeKey);
    getRequest.onsuccess=function(e) {
        var result =getRequest.result;
        console.dir(result);
    }
    var request=objectStore.delete(removeKey);
    request.onsuccess = function(e) {
        console.log("success delete record!");
    };
    request.onerror = function(e) {
        console.log("Error delete record:", e);
    };
    //隐藏要删除的元素
    document.getElementById(removeKey).style.display="none";
}
});
} //init
window.addEventListener("DOMContentLoaded", init, false);

```

WebSocket的完整例子

```
<!DOCTYPE html><html>Web
<head>
  <title>Apache Tomcat
    WebSocket Examples: Chat1</title>
<style type="text/css">
  input#chat {width: 410px}
  #console-container {width: 400px; }
  #console {
    border: 1px solid #CCCCCC;
    border-right-color: #999999;
    border-bottom-color: #999999;
    height: 170px;
    overflow-y: scroll;
    padding: 5px;
    width: 100%;
  }
  #console p {padding: 0;margin: 0;}
</style>
```

[chat2.html](#)

type and press enter to chat

Info: WebSocket connection opened.
* Guest0 has joined.
Guest0: Hello
* Guest1 has joined.
Guest1: Hi
Guest0: How are you?
Guest1: I'm fine

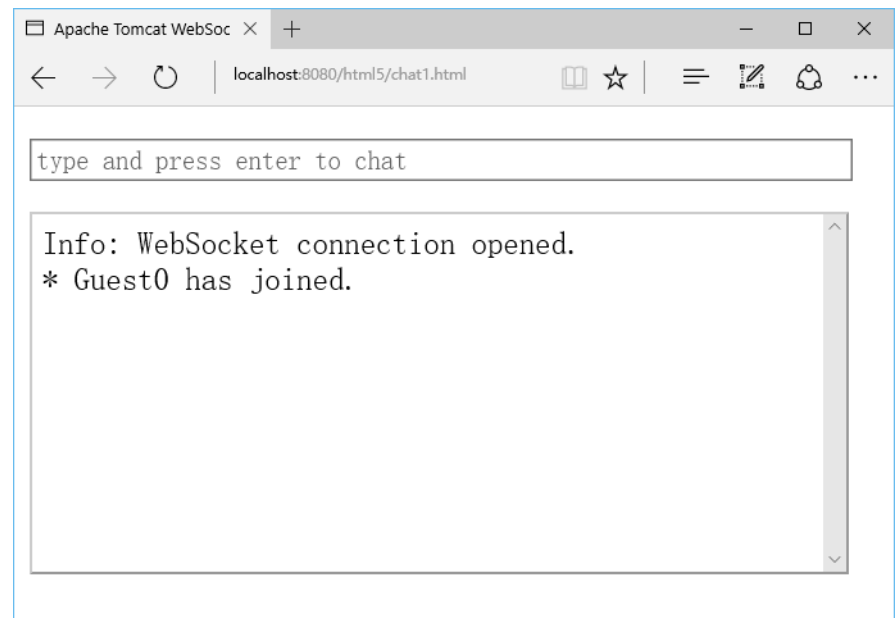
type and press enter to chat

Info: WebSocket connection opened.
* Guest1 has joined.
Guest1: Hi
Guest0: How are you?
Guest1: I'm fine

```

<script type="application/javascript">
    .....
</script>
</head>
<body>
<div class="noscript"><h2 style="color: #ff0000">Seems your browser doesn't
    support Javascript! Websockets rely on Javascript being enabled. Please
    enable Javascript and reload this page!</h2>
</div>
<div>
    <p>
        <input type="text" placeholder="type and press enter to chat" id="chat"/>
    </p>
    <div id="console-container">
        <div id="console"/>
    </div>
</div>
</body>
</html>

```



```

var Chat = {};
Chat.socket = null;
Chat.connect = (function(host) {
    if ('WebSocket' in window) {
        Chat.socket = new WebSocket(host);
    } else if ('MozWebSocket' in window) {
        Chat.socket = new MozWebSocket(host);
    } else {
        Console.log('Error: WebSocket is not supported by this
                    browser. ');
        return;
    }
    Chat.socket.onopen = function () {
        Console.log('Info: WebSocket connection opened. ');
        document.getElementById('chat').onkeydown
            = function(event) {
                if (event.keyCode == 13) {    //按了回车键
                    Chat.sendMessage();
                }
            };
    };
});

```

```

Chat.socket.onclose = function () {
    document.getElementById('chat').onkeydown = null;
    Console.log('Info: WebSocket closed.');
```

```

};
Chat.socket.onmessage = function (message) {
    Console.log(message.data);
};
});
Chat.initialize = function() {
    if (window.location.protocol == 'http:') {
        Chat.connect('ws://' + window.location.host + '/html5/chat');
    } else {
        Chat.connect('wss://' + window.location.host + '/html5/chat');
    }
};
Chat.sendMessage = (function() {
    var message = document.getElementById('chat').value;
    if (message != '') {
        Chat.socket.send(message);
        document.getElementById('chat').value = '';
    }
});

```



```
var Console = {};  
Console.log = (function(message) {  
    var console = document.getElementById('console');  
    var p = document.createElement('p');  
    p.style.wordWrap = 'break-word';  
    p.innerHTML = message;  
    console.appendChild(p);  
    while (console.childNodes.length > 25) {  
        console.removeChild(console.firstChild);  
    }  
    console.scrollTop = console.scrollHeight;  
});  
Chat.initialize();  
document.addEventListener("DOMContentLoaded", function() {  
    var noscripts = document.getElementsByClassName("noscript");  
    for (var i = 0; i < noscripts.length; i++) {  
        noscripts[i].parentNode.removeChild(noscripts[i]);  
    }  
}, false);
```

```

package chat;
import java.io.IOException;           /WEB-INF/classes/chat/ChatExample.class
import java.util.Set;
import java.util.concurrent.CopyOnWriteArraySet;
import java.util.concurrent.atomic.AtomicInteger;
import javax.websocket.OnClose;
import javax.websocket.OnError;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;
import org.apache.juli.logging.Log;
import org.apache.juli.logging.LogFactory;
import util.HTMLFilter;
@ServerEndpoint(value = "/chat")
public class ChatExample {
    private static final Log log = LogFactory.getLog(ChatExample.class);
    private static final String GUEST_PREFIX = "Guest";
    private static final AtomicInteger connectionIds = new AtomicInteger(0);
    private static final Set<ChatExample> connections
        = new CopyOnWriteArraySet<>();

    private final String nickname;
    private Session session;
    public ChatExample() {
        nickname = GUEST_PREFIX + connectionIds.getAndIncrement();
    }
}

```

```

@OnOpen
public void start(Session session) {
    this.session = session;
    connections.add(this);
    String message = String.format("* %s %s", nickname, "has joined.");
    broadcast(message);
}

@OnClose
public void end() {
    connections.remove(this);
    String message = String.format("* %s %s",
        nickname, "has disconnected.");
    broadcast(message);
}

@OnMessage
public void incoming(String message) {
    // Never trust the client
    String filteredMessage = String.format("%s: %s",
        nickname, HTMLFilter.filter(message.toString()));
    broadcast(filteredMessage);
}

```

```

@OnError
public void onError(Throwable t) throws Throwable {
    Log.error("Chat Error: " + t.toString(), t);
}
private static void broadcast(String msg) {
    for (ChatExample client : connections) {
        try {
            synchronized (client) { // 对象同步
                client.session.getBasicRemote().sendText(msg);
            }
        } catch (IOException e) {
            Log.debug("Chat Error: Failed to send message to client", e);
            connections.remove(client);
            try {
                client.session.close();
            } catch (IOException e1) {
                // Ignore
            }
            String message = String.format("* %s %s",
                client.nickname, "has been disconnected.");
            broadcast(message);
        }
    }
}
}

```

[WebSocket session参考](#)

编译命令参考: C:>javac -classpath ./websocket-api.jar;./tomcat-coyte.jar;./tomcat-juli.jar ChatExample.java
 * jar包可以在tomcat下找到

参考地址

- WebSocket:
<http://www.jdon.com/idea/javaee7/websocket.html>
<http://www.open-open.com/lib/view/open1428648292500.html>
<http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/BinaryWebSocket/binaryWebSocket.html>
- Web Worker:
<https://html.spec.whatwg.org/multipage/workers.html>
- W3C
<http://www.w3.org/TR/>