目录

[HTML新特性 1](#_Toc491107428)

[1. 增强型表单（表单2.0） 1](#_Toc491107429)

[2. 新特性表单 2](#_Toc491107430)

[3. 表单新属性 2](#_Toc491107431)

[4. 视频/音频 3](#_Toc491107432)

[5. Canvas 绘图 4](#_Toc491107433)

[6. 位图和矢量图的区别 9](#_Toc491107434)

[7. Canvas和svg的区别 9](#_Toc491107435)

[8. Svg绘图 10](#_Toc491107436)

[9. 地理定位 13](#_Toc491107437)

[10. html5新特性 --拖放API 15](#_Toc491107438)

[11. 新特性 --Web Worker --代码3行 17](#_Toc491107439)

[12. 新特性之九-WebStorage 19](#_Toc491107440)

[13. HTML5新特性之十:WebSocket 22](#_Toc491107441)

# HTML新特性

新语义标签

## 增强型表单（表单2.0）

<**form action="#"**>  
 <**ul**>  
 <**li**>  
 *<!—自带功能\w{1,}@\w{1,}///存在兼容性-->* 邮件地址 :<**input type="email"name="memail"**/><**br**/>  
 url地址:<**input type="url"name="myurl"**/><**br**/>  
 年龄：<**input type="number"name="myage" min="18"max="45"step="2"**/><**br**/>  
 电话: <**input type="tel"name="myrel"** /><**br**/>  
 滑动条：<**input type="range"name="myrel"value="0"** /><**br**/>  
 搜索：<**input type="search"name="mysearch"**/><**br**/>  
 颜色选择器：<**input type="color"name="myrcolor"**/><**br**/>  
 日期：<**input type="date"name="mydate"** /><**br**/>  
 月：<**input type="month"name="mymonth"**/><**br**/>  
 星期：<**input type="week"name="myweek"** /><**br**/>  
 </**li**>  
 <**li**>  
 <**input type="submit"**/>  
 </**li**>  
 </**ul**>  
</**form**>

## 新特性表单

* + **建议列表**

<datalist id="list3">

<option>xx</option> datalist本身不可见

<option>yyy</option>

</datalist>

<input type="text" list="list3"/>

* + **进度条progress**

<progress></progress> 左右晃动进度条

<progress value="0.7"></progress> 具有指定进度值的进度条（用定时器改变value进度）

* + **Meter:度量衡,刻度尺,用于标识一个值所在范围:**

不可接受(红色),可以接受(黄色)非常优秀(绿)

<meter min="可取最小值" max="可取最大值"

low="合理下限" high="合理上限"

optimum="最佳值" value="当前实际值" ></meter>

output:输出:语义标签，没有任何外观样式，样式等同span

购买数量:<input type="number" value="2">

小计:<output>3.50</output>

## 表单新属性

<input ?>

h4:name;id;type;class;title;style;value;disabled;readonly;checked

h5:

(1)placeholder: 占位字符

(2)autofocus: 自动获取输入焦点

(3)multiple: 允许输入框中出现多个输入(用逗号分隔) 如:邮箱

(4)form: 用于把输出域放置到form外部

=====输入验证相关的新属性

密码<input type="password" name="upwd" placeholder="请输入密码" form="f5"/>

<form action="#" id="f5">

(5)required 必填项:内容不能为空

(6)maxlength 指定字符串最大长度

(7)minlength 指定字符串最小长度

(8)max 指定数字的最大值

(9)min 指定数字的最小值

(10)pattern 指输入必须符合正则表达式

## 视频/音频

* **视频**

<video src="res/birds.mp4"></video>

<video>

<source src="res/brids.mp4" />

<source src="res/brids.ogg" />

<source src="res/brids.webm" />

您的浏览器不支持 video播放

</video>

它本身是一个300\*150的inline-block元素

**成员属性:（video）**

autoplay: false 是否自动播放

controls:false/ controls:”controls” 是否显示播放控件

loop:false 是否循环播放

muted:false 是否静音

poster:"" 在播放第一帧之前显示海报

preload 视频的预加载策略

auto:预加载视频和元数据和缓冲一定时长

metadata:仅预加载视频元素数据

(尺寸，时长,第一帧内容),没有视频缓冲

none：不预加载任何数据

**----JS对象属性(#vi.)-----**

currentTime: 当前播放的时长

duration: 总时长

paused:true 当前视频是否处理暂停状态

volume:1 当前音量

playbakRate:1 回放速度 大于1表示快放,小于1表示慢放

**成员方式:**  
play() 播放视频  
pause() 暂停播放  
**成员事件**  
onplay 当视频开始播放时触发的事件  
onpause 当视频开始暂停时触发事件

<h1>h5新特性--视频播放</h1>

<video src="res/birds.mp4" id="v3" controls poster="res/play.png" preload="metadata">

</video>

<script>

//调整音量

v3.volume=1;

//回放速度

//v3.playbackRate=0.5;//半倍速 更慢

v3.playbackRate=3;//3倍速播放 更快

</script>

* **音频**

<audio src="res/bg.mp3"></audio>

<audio>

<source src="reg/bg.mp3">

<source src="reg/bg.wav">

</audio>

它默认是一个300\*30的inline-block;但若没有controls属性不显示.

## Canvas 绘图

网页中实时走势图、统计图、网页游戏、地图应用都要使用到绘图技术，有三种绘图技术:

(1)SVG绘图:2D**矢量绘图**技术，2000出现，后纳入h5标准

(2)Canvas绘图:2D**位图绘图**技术，h5提出绘图技术

(3)WebGL绘图:3D绘图技术，尚未纳入H5标准.

canvas标签在浏览器默认300\*150的inline-block.

画布宽和高只能使用HTML/js属性来赋值，

***不能使用CSS样式赋值***.

* **创建画布、矩形、描边**

**var ctx = canvas.getContext("2d");//得到画布上画笔对象**

ctx.lineWidth = 1; //描边宽度(边线)

ctx.fillStyle = "#000"; //填充样式

ctx.strokeStyle = "#000"; //描边样式(边线)

ctx.fillRect(x,y,w,h); //填充一个矩形

ctx.strokeRect(x,y,w,h); //描边矩形

ctx.clearRect(x,y,w,h); //清除一个矩形范围内所有绘图

<h1>h5新特性--canvas绘图</h1>

<canvas id="c3" width="500" height="400">

您的浏览器不支持canvas标签

</canvas>

<script>

//1:通过画布获取画笔

var ctx = c3.getContext("2d");

//2:设置画笔样式

//在画布左上角画一个填充矩形

ctx.fillStyle="#f00";

//3:填充矩形

ctx.fillRect(0,0,100,100);

</script>

* **绘制45度角移动的矩形**

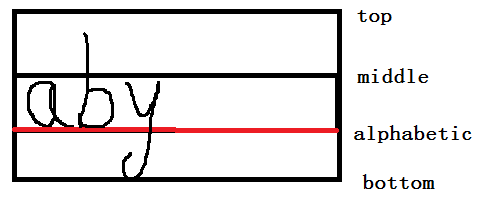
<**h1**>canvas绘制45度移动的矩形 </**h1**>  
<**canvas id="c4" width="500"height="500"**></**canvas**>  
<**script**>  
 **var *ctx***=c4.getContext(**"2d"**);  
 ***ctx***.**fillStyle**=**"#00f"**;

**var *x***=0;  
 **var *y***=0;  
 **var *xDirection***=1;  
 **var *yDirection***=1;  
 **var *t***=setInterval(**function**(){  
 ***ctx***.clearRect(0,0,100,80);  
 ***ctx***.clearRect(***x***,***y***,100,80);  
 ***x***+=***xDirection***;  
 ***y***+=**Math**.tan(**Math**.**PI**/4)\****yDirection***;  
 ***ctx***.fillRect(***x***,***y***,100,80);  
 **if**(***x***>=400){  
 ***xDirection***=-1;  
 }**else if**(***x***<=0){  
 ***xDirection***=1;  
 }  
 **if**(***y***>=500-80){  
 ***yDirection***=-1  
 }**else if**(***y***<=0){  
 ***yDirection***=1  
 }  
 },10)  
</**script**>

* **计算移动角度的公式**

***y***+=**Math**.tan(**Math**.**PI**/4)\****yDirection***;

* **Canvas绘制文本；**



一段文字的定位点在其文本基线的起点

ctx.textBaseLine = "alphabetic";//文本基线

ctx.font = "12px sans-serif";//文本大小和字体

ctx.fillText(str,x,y); //填充文本

ctx.strokeText(str,x,y); //描边文本

ctx.measureText(str); //基于当前文本测量,返回对象{width:x}

<**script**>  
 **var *ctx***=c3.getContext(**"2d"**);  
 ***ctx***.**textBaseline**=**"top"**;  
 ***ctx***.**font**=**"30px SimHei"**;  
 **var *str***=**"tarena"**;  
 **var *obj***=***ctx***.measureText(***str***);  
 ***ctx***.fillText(***str***,0,0);

* **Canvas绘制渐变对象**

线性渐变:linearGradient

径向渐变:

var g = ctx.createLinearGradient(x1,y1,x2,y2);

x1,y1 起点坐标

x2,y2 终点坐标

g.addColorStop(0,"red");

g.addColorStop(1,"green")

ctx.fillStyle = g;

ctx.strokeStyle = g;

**var *ctx***=c3.getContext(**"2d"**); *//canvas的id=c3***var *g***=***ctx***.createLinearGradient(0,0,500,400);  
***g***.addColorStop(0,**"red"**);  
***g***.addColorStop(1,**"#0f0"**);  
***ctx***.**fillStyle**=***g***;  
***ctx***.fillRect(0,0,500,50);  
***ctx***.fillText(**"123456479895464646541321346413"**,0,70);

* **Canvas进行绘图--路径（直线和弧度图形）**

Path:类似于PS中"钢笔工具",由多个坐标点组成任意形状，路径不可见，可用于"描边","填充","裁剪"

ctx.beginPath(); 开始一条新路径

ctx.closePath(); 闭合当前路径

ctx.moveTo(x,y); 移动到指定点

ctx.lineTo(x,y); 从当前的点到指定点画直线

ctx.arc(cx,cy,r,start,end); 绘制圆拱路径

cx,cy 圆心

r 半径

start 启始角度

end 结束角度

ctx.stroke(); 对当前路径描边

ctx.fill(); 对当前路径填充

ctx.clip(); 对当前路径裁剪

**var *ctx***=c2.getContext(**"2d"**);  
 ***ctx***.beginPath();  
*// ctx.moveTo(50,50);  
// ctx.lineTo(50,350);  
// ctx.lineTo(450,350);  
// ctx.lineTo(450,50);  
// ctx.stroke();* ***ctx***.arc(250,200,100,**180\*Math.PI/180**,360\***Math**.**PI**/180);  
 ***ctx***.stroke();

* **计算绘制弧度的公式:（角度转化成弧度）**

**90\*Math.PI/180**, //90为需要转换的角度，Math.PI/180是转换的固定值

* **canvas绘图--图像(图片)**

Canvas属于客户端技术，**图片在服务器**中，所以浏览器必须先下载绘制图片，且等待图片异步加载完成.

var p3 = new Image();

p3.src = "img/p3.jpg"; //向服务器发出请求并且获取图片

p3.onload = function(){ //onload 图片下载完成

console.log(p3.width); //正确数据

ctx.drawImage(p3,x,y); //原始大小绘图

ctx.drawImage(p3,x,y,w,h); //拉伸图片

}

<**canvas id="c3" width="500" height="400"**></**canvas**>  
<**script**>  
 **var *ctx*** = c3.getContext(**"2d"**);  
 **var *p3*** = **new *Image***();  
 ***p3***.src = **"img/p3.png"**;  
 ***p3***.onload = **function**(){  
 ***console***.log(***p3***.width);  
 ***ctx***.drawImage(***p3***,0,0);*//左上角* ***ctx***.drawImage(***p3***,500-***p3***.width,0);*//右上角* }

</**script**>

* **使用canvas绘图时进行变形操作（旋转、原点平移）**

canvas绘图中有变形技术,可以针对某一个图像/绘制过程变形

rotate/scale;translate

ctx.rotate(弧度); 旋转绘图上下文对象(即画笔),轴点是画布**原点**

ctx.translate(x,y); 将整个画布原点平移到指定点

ctx.save(); 保存画笔当前所有变形值

ctx.restore(); 恢复画笔状态到最近一次保存

练习:在画布的左上角画一个绕自己为中心旋转飞机1;在同一个画布的右上角画一个旋转速度是飞机1, 2倍速的飞机2.

飞机1

飞机2

ctx.translate(100,50); :平移

ctx.rotate(10); :旋转 ctx.rotate(20);

ctx.drawImage(p3,..) :绘制

ctx.rotate(-10); :逆向旋转

ctx.translate(-100,-50):逆向平移

提示:每个小飞机"平移+旋转+绘制+逆向旋转+逆向平移"才不会影响后续飞机的绘制.

<**canvas id="c3" width="500" height="400"**></**canvas**>  
 <**script**>  
 **var *ctx*** = c3.getContext(**"2d"**);  
 **var *p3*** = **new *Image***();  
 ***p3***.src = **"img/p3.png"**;  
 ***p3***.onload = **function**(){  
 *//1:创建二个变量分别保存旋转角色* **var** deg1 = 10;*//第一架飞机旋转10度* **var** deg2 = 20;*//第二架飞机旋转20度  
 //2:创建定时器* **var** t = setInterval(**function**(){  
 *//3:绘制第一架飞机  
 //3.0:保存第一架飞机状态(圆点0,0,旋转角度0)* ***ctx***.save();  
 *//3.1:平移圆点* ***ctx***.translate(100,50);  
 *//3.2:旋转* ***ctx***.rotate(deg1\*Math.PI/180);  
 *//3.3:绘制* ***ctx***.drawImage(***p3***,-100,-50);  
 *//3.4:逆向旋转  
 //ctx.rotate(-deg1\*Math.PI/180);  
 //3.5:逆向平移  
 //ctx.translate(-100,-50);* ***ctx***.restore();  
 *//3.6:旋转角度不进行累加，我们只手工累加* deg1 += 10;  
*// //4:绘制第二架飞机  
 //4.0 保存第二架飞机状态(圆点,旋转角度)* ***ctx***.save();  
 *//4.1:平移圆点* ***ctx***.translate(100,350);  
 *//4.2:旋转 14:35--14:40* ***ctx***.rotate(deg2\*Math.PI/180);  
 *//4.3:绘制* ***ctx***.drawImage(***p3***,-100,-50);  
 *//4.4:逆向旋转  
 //ctx.rotate(-deg2\*Math.PI/180);  
 //4.5:逆向平移  
 //ctx.translate(-100,-350);* ***ctx***.restore();  
 *//4.6:由于逆向旋转，角度不再累加，手工累加* deg2 += 20;  
 },50);  
 }  
 </**script**>

## 位图和矢量图的区别

**位图**:由一个一个像素点组成，每个点各有自己的颜色，色彩细腻，但放大会失真.

**矢量图**:由一个一个线条组成，每个线条可以指定颜色，可以无限缩放，但细节不够丰富.

## Canvas和svg的区别

|  |  |  |
| --- | --- | --- |
|  | canvas绘图 | svg绘图 |
| 类型 | 2D位置 | 2D矢量图 |
| 如何绘图 | 使用JS代码绘图 | 使用标签绘图 |
| 事件绑定 | 每个图形不是元素，无法直接绑定事件 | 每个图形都是元素，可以直接绑定事件 |
| 应用场合 | 统计图,游戏 | 统计图,图标,地图 |

## Svg绘图

**SVG技术在H5出现之后使用方法**

直接在html5文档中，书写svg标签即可.

<svg></svg> 本身是300\*150的inline-block;

* **<rect> 使用SVG标签绘制矩形**

<rect width="" height="" x="" y="" fill="" fill-opacity=""

stroke="" stroke-width="" stroke-opacity=""></rect>

* **<circle> 使用SVG标签绘制圆形**

<circle r="" cx="" cy="" fill="" fill-opacity="" stroke="" storke-opacity="">

<**h1**>在h5标准中使用svg绘制图形--圆形</**h1**>  
<**svg width="500" height="400" id="s3"**>  
</**svg**>  
<**script**>  
 *//1:准备二个函数  
 //rn 返回指定范围内的随机整数  
 //rc 返回指定范围内随机颜色* **function** *rn*(min,max){  
 **return** Math.floor(  
 Math.random()\*(max-min)+min);  
 }  
 **function** *rc*(min,max){  
 **var** r=*rn*(min,max);  
 **var** g=*rn*(min,max);  
 **var** b=*rn*(min,max);  
 **return** `rgb(${r},${g},${b})`;  
 }  
 *//2:创建循环30次，创建30圆* **for**(**var *i***=0;***i***<30;***i***++){  
 **var *c*** = ***document***.createElementNS(  
 **"http://www.w3.org/2000/svg"**,  
 **"circle"**);*//17:00--17:05* ***c***.setAttribute(**"r"**,*rn*(10,80));  
 ***c***.setAttribute(**"cx"**,*rn*(0,500));  
 ***c***.setAttribute(**"cy"**,*rn*(0,400));  
 ***c***.setAttribute(**"fill"**,*rc*(0,255));  
 ***c***.setAttribute(**"fill-opacity"**,Math.random());  
 *//为每一个元素绑定点击事件* ***c***.onclick = **function**(){  
 *//1:保存this* **var** that = **this**;  
 *//2:防止当前元素再次点击* that.onclick = **null**;  
 *//3:启动定时器，让圆形变大变淡* **var** t = setInterval(**function**(){  
 *//变大* **var** r = that.getAttribute(**"r"**);  
 r \*= 1.05;  
 that.setAttribute(**"r"**,r);  
 *//变淡* **var** p = that.getAttribute(**"fill-opacity"**);  
 p \*= 0.9;  
 that.setAttribute(**"fill-opacity"**,p);  
 **if**(p<0.0001){  
 clearInterval(t);  
 s3.removeChild(that);  
 }  
 },30);  
 };  
 s3.appendChild(***c***);  
 }  
</**script**>

* **<ellipse> 使用SVG标签绘制椭圆形**

<ellipse rx="" ry="" cx="" cy=""></ellipse>

rx:水平半径

ry:垂直半径

cx/cy：圆心距离原点的距离

* **<line> 使用SVG标签绘制直线**

<line x1="" y1="" x2="" y2="" stroke="" stroke-width="" stroke-linecap="butt/square/round">



<**h1**>在h5标准中使用svg绘制图形--椭圆形</**h1**>  
<**svg width="500" height="400" id="s3"**>  
 *<!--group:小组:本身不可见，保存共有属性-->* <**g stroke="#f00" stroke-width="50" stroke-linecap="round"**>  
 <**line x1="100" y1="100" x2="110" y2="100"**></**line**>  
 <**line x1="170" y1="100" x2="370" y2="100"**></**line**>  
 <**line x1="100" y1="180" x2="110" y2="180"**></**line**>  
 <**line x1="170" y1="180" x2="370" y2="180"**></**line**>  
 <**line x1="100" y1="260" x2="110" y2="260"**></**line**>  
 <**line x1="170" y1="260" x2="370" y2="260"**></**line**>  
 </**g**>  
</**svg**>

* **SVG 绘图 -- 折线**

一条折线上可以有任意多个连接的点

<polyline points="50,50 100,10 ..." fill="transparent" stroke="#000"> </polyline>

<**polyline points="50,100 150,100 75,175 100,45 125,175 50,100"   
 fill="transparent" stroke="#000" stroke-width="5"**></**polyline**>

* **SVG 绘图 -- 多边形**

<polygon points="50,50 100,10 ..." fill="transparent" stroke="#000"> </polygon>

* **SVG 绘图 -- 文本**

<text font-sze="" alignment-baseline="" fill="" stroke="" x="" y="">

你的文本内容

</text>

<**text font-size="50" alignment-baseline="alphabetic"  
 fill="transparent" stroke="#000" x="10" y="50"**>tarena</**text**>

* **SVG 绘图 -- 图像**

<image xlink:href="x.png" width="" height="" x="" y="">

<**image xlink:href="img/p3.png" x="100" y="100" width="200" height="100"**></**image**>

* **SVG 绘图 -- 渐变对象**

<defs> 定义特效对象:渐变对象属于一种特效对象

<linearGradient id="g3" x1="" y1="" x2="" y2="">

<stop offset="" stop-color="" stop-opacity="" />

<stop offset="" stop-color="" stop-opacity="" />

</ linearGradient>

<defs>

<\* fill="url(#g3)" stroke="url(#g3)" >

*<!--渐变对象属于一种特效对象需要单独定义-->*<**defs**>  
 <**linearGradient id="g3" x1="0%" y1="0%" x2="0%" y2="100%"**>  
 <**stop offset="0%" stop-color="red"**></**stop**>  
 <**stop offset="17%" stop-color="orange"**></**stop**>  
 <**stop offset="34%" stop-color="yellow"**></**stop**>  
 <**stop offset="51%" stop-color="green"**></**stop**>  
 <**stop offset="68%" stop-color="cyan"**></**stop**>  
 <**stop offset="85%" stop-color="blue"**></**stop**>  
 <**stop offset="100%" stop-color="purple"**></**stop**>  
 </**linearGradient**>  
</**defs**>  
<**rect width="50" height="100" x="0" y="0" fill="url(#g3)"**></**rect**>  
<**rect width="50" height="100" x="100" y="0" fill="url(#g3)"**></**rect**>  
<**rect width="50" height="100" x="200" y="0" fill="url(#g3)"**></**rect**>

* **SVG 绘图 -- 滤镜(高斯模糊滤镜)**

<defs> 定义滤镜属于一种特效对象

<filter id="f2">

<feGaussianBlur stdDeviation="3" />

</filter>

</defs>

<\* filter="url(#f2)">

<**defs**>  
 <**filter id="f2"**>  
 <**feGaussianBlur stdDeviation="2"**></**feGaussianBlur**>  
 </**filter**>  
 <**filter id="f3"**>  
 <**feGaussianBlur stdDeviation="6"**></**feGaussianBlur**>  
 </**filter**>  
</**defs**>  
<**text font-size="70" y="100"**> 达内科技2017</**text**>  
<**text font-size="70" y="200" filter="url(#f2)"**> 达内科技2017</**text**>  
<**text font-size="70" y="300" filter="url(#f3)"**> 达内科技2017</**text**>

## 地理定位

Geolocation:地理定位，使用JS获取当前浏览器的地理坐标(经度,纬度,海拨,速度)数据,用于实现LBS应用(Location Based Service)如 饿了么,嘀嘀

**手机浏览器如何获得定位信息**

(1)首选手机中的GPS芯片与卫星通信，定位精度在米

(2)次选手通信基站定位获取，定位精度在公里

**PC浏览器如何获得定位信息**

通过ip地址进行反向解析，定位精度取决于IP地址库的大小

HTML5中提供了一个新的对象，用于获取当前浏览器定位信息

window.navigator.geolocation{

getCurrentPosition:fn, 获得当前定位信息

watchPosition:fn, 监控数据的改变

clearWatch:fn 取消监视

}

获得客户的定位信息

window.navigator.geolocation.getCurrentPostion(

(pos)=>{pos.coords.latitude;pos.coords.longitude}

);

<**button id="btn1"**>获取位置</**button**>  
<**script**>  
 btn1.onclick = **function**(){  
 *//console.log(window.navigator.geolocation);* ***navigator***.**geolocation**.getCurrentPosition(*succCB*,*errCB*);  
 }  
 **function** *succCB*(pos){  
 ***console***.log(**"成功获取到定位数据"**);  
 ***console***.log(**"定位的时间"**+pos.**timestamp**);  
 ***console***.log(**"纬度"**+pos.coords.latitude);  
 ***console***.log(**"经度"**+pos.coords.longitude);  
 ***console***.log(**"海拨"**+pos.coords.altitude);  
 ***console***.log(**"速度"**+pos.coords.speed);  
 }  
 **function** *errCB*(err){  
 ***console***.log(**"获取到定位数据失败"**);  
 ***console***.log(**"错误编号"**+err.**code**);  
 ***console***.log(**"错误内容"**+err.**message**);  
 }  
</**script**>

* ***扩展小知识:百度地图开发者***

(1)注册百度开发者帐号

http://lbsyun.baidu.com/

(2)创建一个网站:登录百度地图，为网站申请求地图 AcessKey

(3)在自己网页中嵌入百度地图提供API，

http://lbsyun.baidu.com/index.php?title=jspopular

a:创建地图对象

var map = new BMap.Map("container");

b:创建坐标点

var pointer = new BMap.Point(经度,纬度);

c:以指定为中心并且缩放1-19

map.centerAndZoom(pointer,17)

<**style type="text/css"**>  
 **html**{**height**:100%}  
 **body**{**height**:100%;**margin**:0**px**;**padding**:0**px**}  
 **#container**{**height**:100%}  
 </**style**>  
 <**script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=u70A5pnNrRtRT1XAgwM5jL2YIVj1Gv97"**>  
 *//v2.0版本的引用方式：src="http://api.map.baidu.com/api?v=2.0&ak=您的密钥"  
 //v1.4版本及以前版本的引用方式：src="http://api.map.baidu.com/api?v=1.4&key=您的密钥&callback=initialize"* </**script**>  
</**head**>  
<**body**>  
<**div id="container"**></**div**>  
<**script type="text/javascript"**>  
 **var *map*** = **new** BMap.Map(**"container"**); *// 创建地图实例* **var *point*** = **new** BMap.Point(116.300984,39.916068); *// 创建点坐标* ***map***.centerAndZoom(***point***, 19); *// 初始化地图，设置中心点坐标和地图级别  
 //启用滚轮缩放* ***map***.enableScrollWheelZoom(**true**);  
 *//添加导航控件* ***map***.addControl(**new** BMap.NavigationControl());  
 *//添加缩放控件* ***map***.addControl(**new** BMap.ScaleControl());  
 *//添加概览图控件* ***map***.addControl(**new** BMap.OverviewMapControl());  
 *//添加地图类型控件* ***map***.addControl(**new** BMap.MapTypeControl());  
</**script**>

## html5新特性 --拖放API

Drag&Drop: 拖放和释放

拖动的源对象:可以触发三个事件

dragstart:拖动开始

drag: 拖动中

dragend: 拖动结束

拖动的目标对象:可以触发四个事件

dragenter: 拖动着进入

dragover: 拖动着悬停在上方

dragleave: 拖动着离开

drop: 在上方释放

拖动过程1:dragenter\*1+dragover\*n+dragleave\*1

拖动过程2:dragenter\*1+dragover\*n+drop\*1

**注意:**必须阻止dragover的**默认行为**,drop才可能触发

<**h1**>请选择此次出征的英雄</**h1**>  
<**div id="chosen"**>  
 *<!--目标区域-->* <**img src="img/p0.png" id="p0"**/>  
</**div**>  
<**div id="list"**>  
 *<!--列表区域-->* <**img src="img/p3.png" id="p3"**/>  
 <**img src="img/p4.png" id="p4"**/>  
 <**img src="img/p5.png" id="p5"**/>  
</**div**>  
<**script**>  
 *//1:拖动某架飞机到目标区域,则上方显示该飞机  
 //拖动源对象: #list>img  
 //拖动目标对象: #chosen  
 //2:全局变量，保存被拖动的飞机id* **var *dragId*** = **null**;  
 *//3:获取所有源对象并且绑定拖动三个事件* **var *lists*** = ***document***.querySelectorAll(**"#list img"**);  
 **for**(**var *i***=0;***i***<***lists***.length;***i***++){  
 **var *p*** = ***lists***[***i***];  
 ***p***.ondragstart = **function**(e){  
 ***dragId*** = **this**.***id***;  
 }  
 ***p***.ondrag = **function**(e){}  
 ***p***.ondragend = **function**(e){}  
 }  
 *//4:拖动目标对象触发的四个事件* chosen.ondragenter = **function**(){}  
 chosen.ondragover = **function**(e){  
 e.preventDefault()}  
 chosen.ondragleave = **function**(){}  
 chosen.ondrop = **function**(){  
 *//5:问号飞机隐藏* p0.style.**display** = **"none"**;  
 **var** p = ***document***.getElementById(***dragId***);  
 chosen.appendChild(p);  
 *//6:拖动另一架飞机到目标区域，  
 // 显示该飞机，之前选中飞机重回下方列表  
 //若如此时#chosend子元素数量己经为3个,  
 //则必须p0后的那个兄弟元素重新回到#list* **if**(chosen.children.**length**>=3){  
 **var** n = p0.**nextElementSibling**;  
 list.appendChild(n);  
 }  
 }  
 *//7:若拖动选中飞机回到下方列表区域  
 //上方重新显示问号飞机  
 //拖动:#chose中的飞机到#list上方  
 //拖动源对象 #chose img  
 //拖动目标对象 #list  
 //当前需要拖动飞机id保存列表中* list.ondragenter = **function**(){}  
 list.ondragover = **function**(e){  
 e.preventDefault();  
 }  
 list.ondragleave = **function**(){}  
 list.ondrop = **function**(){  
 **if**(***dragId*** !==**"p0"**){*//9:55--10:00  
 //把非p0飞机移动list列表中* **var** p = ***document***.getElementById(***dragId***);  
 list.appendChild(p);  
 *//显示p0* p0.style.**display** = **"block"**;  
 }  
 }  
</**script**>

## 新特性 --Web Worker --代码3行

程序:program:一堆代码保存在外储中，指被cpu执行.

进程:process/Task:指程序被操作系统OS调用内存,分配执行空间,随时供CPU调度执行.

线程:Thread:线程进程内部执行代码基本单位

**进程和线程**

(1)进程是操作系统分配内存的基本单位

(2)线程是CPU执行代码的基本单位

(3)线程必须处理某个进程内部

(4)一个进程内必须至少有一个线程，也可以有多个

(5)一个操作系统可能同时存在几千个线程，它们是"并发执行的"--宏观上看同时执行，微观上看是依次循环执行

**Chrome浏览器中的线程模型**

一个Chrome进程内，至少有6个线程，可以"同时/并发"向web服务器发起http请求，以获得所需的资源---资源请求线程

还有一个线程负责将所有内容绘制到浏览器页面中--UI主线程--不允许多个线程同时绘图，防止内容布局错乱.

**Worker线程的致命缺陷**  
浏览器不允许Woker线程操作任何的DOM&BOM对象  
原因:浏览器只允许UI主线程操作DOM&BOM若多个线程同时都可操作DOM结构，页面将混乱.  
所以类似:jQuery脚本决不能用Worker来加载执行.

**Woker线程可以给UI主线程发送数据消息**

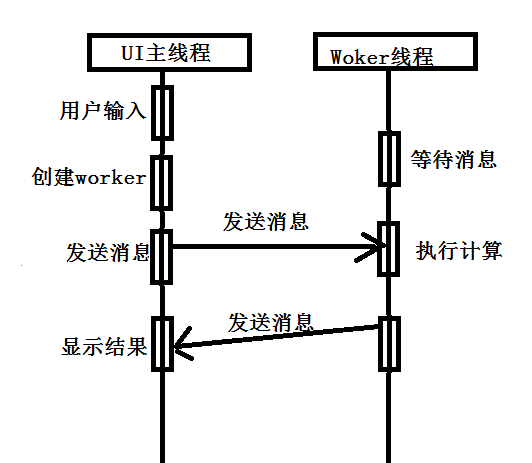
Worker线程:

postMessage(stringMsg);

UI线程:

var w1 = new Worker("01.js");

w1.onmessage = function(e){e.data}

 **UI线程可以给Worker线程发数据消息**

UI线程:

var w1= new Worker("01.js");

w1.postMessage(stringMsg);

Worker线程:

onmessage = function(e){e.data}

**2.html代码**

<**h1**>webworker 练习</**h1**>  
 <**input type="text" id="inputNum"**/>  
 <**button id="bt"**>开始计算</**button**>  
 <**div id="result"**>计算中...</**div**>  
<**script**>  
 *//1:获取用户按钮bt绑定点击事件  
 //11:00--11:05* bt.onclick = **function**() {  
 *//2在事件内部创建Worker线程将用户输入数据发送过去* **var** n = inputNum.value;  
 **var** w2 = **new** Worker(**"02.js"**);  
 *//ui线程发送数据给worker线程* w2.postMessage(n);  
 *//ui线程接收worker线程来的数据* w2.onmessage =**function**(e){  
 ***result***.**innerHTML** = e.data;  
 }  
 }  
</**script**>

**2.js代码**

*//等待接收消息****onmessage*** = **function**(e){  
 ***console***.log(**"UI主线程发来的数据:"**+e.data);  
 *//开始计算  
 //1:将UI主线程发来数据转换整型* **var** n = *parseInt*(e.data);  
 *//2:累加* **var** sum = 0;  
 **for**(**var** i=1;i<=n;i++){  
 sum += i;  
 }  
 ***console***.log(**"结果:"**+sum);  
 *//Worker将要把计算结果转递UI主线程  
 postMessage*(sum);  
}

## 新特性之九-WebStorage

在浏览器中存储当前用户专有的数据:访问历史、内容定制、样式定制...

**在客户端储数据可以使用的技术**

(1)**Cookie**技术:浏览器兼容性好；不能超过4KB，操作复杂

(2)Flash存储:依赖于Flash播放器

(3)**H5 WebStorage:不能超过8MB,操作简单**

(4)IndexedDB:可存储大量数据，还不是标准技术

|  |
| --- |
| Sesssion:会话:浏览器从打开某个网站的第一个页面开始(会话 开始)，中间可能打开很多页面(会话中间操作)，直到关闭浏览器整个过程称为"会话<浏览器与web服务器的一次会话>"; |

WebStorage技术中，浏览器为用户提供了两个对象

**(1)window.sessionStorage 类数组对象，会话级数据存储**

在浏览器进程所分得"内存"存储着一次web会话可用的数据，

可供此次会话所有页面共同使用;浏览器一旦关闭就消失了。

作用:在同一个会话中所有页面共享数据: 登录用户名/用户编号

* + sessionStorage[key]=value; //保存一个数据
  + sessionStorage.setItem(key,val); //保存一个数据
  + var v = sessionStorage[key]; //读取一个数据value
  + var v = sessionStorage.getItem(key);//读取一个数据 value
  + sessionStorage.removeItem(key); //删除一个数据
  + sessionStorage.clear(); //清除所有的数据
  + sessionStorage.length //数据的数量
  + sessionStorage.key(i); //读取第i个key
* **index.html**

<**div id="welcome"**>  
 <**a href="06\_login.html"**>请登录</**a**>  
</**div**>  
<**script**>  
 *//1:获取当前会话中是否存储 loginName* **var *n*** = ***sessionStorage***.getItem(**"loginName"**);  
 *//2:如果值存在 <登录成功>* **if**(***n***) {  
 *//3:修改welcome中间数据:  
 // 欢迎回来 <a href="">退出登录</a>* welcome.**innerHTML** = `  
 欢迎回来 ${***n***} <a href=**"07\_logout.html"**>退出登录</a>`;  
 }  
</**script**>

* **Login.html**

<**form action="#"**>  
 用户名 <**input type="text" id="uname"**/> <**br**/>  
 密码 <**input type="text" id="upwd"**/>  
 <**br**/>  
 <**input type="button" id="btn" value="提交..."**/>  
</**form**>  
<**script**>  
 *//1:获取提交按钮绑定点击事件* ***btn***.onclick = **function**() {  
 *//2:获取用户名和密码* **var** n = ***uname***.value;  
 **var** p = upwd.value;  
 *//3:发起异步AJAX请求，并且获取返回数据  
 // 判断登录是否成功  
 //4:保存此次登录用户名--会话级存储* ***sessionStorage***.setItem(**"loginName"**,n);  
 *//5:3S自动跳转* alert(**"登录成功，3s后将自动跳转回首页"**);  
 *setTimeout*(**function**(){  
 ***location***.href = **"05\_index.html"**;  
 },3000);  
 }  
</**script**>

* Loginout

<**h1**>退出登录</**h1**>  
<**script**>  
 *//1：清除会话级数据 loginName* ***sessionStorage***.removeItem(**"loginName"**)  
 *//2: 3S后跳转回 05\_index.html  
 setTimeout*(**function**(){  
 ***location***.href = **"05\_index.html"**;  
 },3000); *//14:40--14:42*</**script**>

**(2)window.localStorage**类数组对象,本地存储(**跨**会话级)

在浏览器的所能管理的外存(硬盘)中存储着用户的浏览数据，可供

此次会话以即后续会话中的页面共同使用。即使浏览器关闭也不会消失-永久存在.作用：当前客户端所有对应的所有会话中共享的数据。"网站访问次数".

* + localStorage [key]=value; //保存一个数据
  + localStorage.setItem(key,val); //保存一个数据
  + var v = localStorage [key]; //读取一个数据value
  + var v = localStorage.getItem(key);//读取一个数据 value
  + localStorage.removeItem(key); //删除一个数据
  + localStorage.clear(); //清除所有的数据
  + localStorage.length //数据的数量
  + localStorage.key(i); //读取第i个key

localStorage中若数据发生修改,会触发事件 **window.onstorage**事件,

,可以监听此事件，实现监视localStorage数据改变的目的,用于在一个窗口监视其它的窗口对localStorage数据修改--不能监视sessionStroage数据修改.

<**h1**>用户中心</**h1**>  
<**script**>  
 *//1:读取之前曾经选中主题样式名,直接应用body* **var *t*** = ***localStorage***.getItem(**"userTheme"**);  
 **if**(***t***){  
 ***document***.body.className = ***t***;  
 }  
 *//2:监视localStorage数据改变事件  
 //onstorage  
 //16:00--16:05 调错/首页自动应用样式* ***window***.onstorage = **function**() {  
 *//重新再次读取选中主题,再次为body修改样式* ***console***.log(**"localStroage数据改变啦!"**);  
 **var** t = ***localStorage***.getItem(**"userTheme"**);  
 **if**(t){  
 ***document***.body.className = t;  
 }  
 }  
</**script**>

## HTML5新特性之十:WebSocket

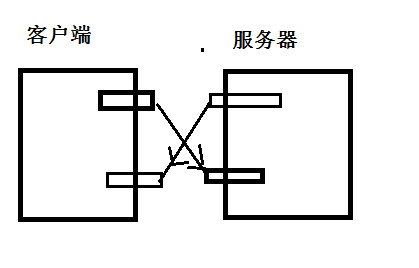
HTTP协议:属于"请求-响应"模型,只有客户端发起请求消息,服务器才会返回响应消息,没有请求没有响应.

有些场景中，此模型力不从心:"实时走势应用"

解决方案:长轮询/心跳请求--定时器+AJAX--请求过于频繁，服务器压力过大；不够频繁，客户数据延迟较大.

WebSocket协议:属于"广播-收听"模型,客户端连接到服务器就不再断开，永久连接,双方随时向对方发送消息,且**全双工**不对等发送.

WS协议:在实时间走势应用，有特别优势



WS协议的应用程序分为客户端程序和服务器端程序

WS服务器应用:

监听指定端口,接客户请求，向对方发送消息,并接收消息;

可以使用php/java/nodejs等语言编写

WS客户端应用:

主动发起连接请求，保持永久的连接,向对方发送消息，并接收消息。可以php/java/node.js/html5

注意事项:nodejs中没有提ws模块,必须使用npm下载第三方模块;

npm i ws

**使用:HTML5创建WS协议的客户端应用**

1:连接到WS服务器

var socket = new WebSocket("ws://127.0.0.1:9001");

2:向服务器发消息

socket.send(stringMsg)

3:接收服务器发来的消息

socket.onmessage = function(e){

e.data;//消息内容

}

4:断开到WS服务器的连接

socket.close();

* 服务器

**const *ws*** = *require*(**"ws"**);  
*//2:创建ws服务器,并且指定监听端口***var *server*** = **new *ws***.Server({**port**:9001});  
*//3:为ws服务器绑定事件 connection  
// 每一个客户端连接都会触发此事件****server***.on(**"connection"**,(socket)=>{  
 *//4:每隔1s不停向客户发送数据 send();* **var** counter = 0;  
 **var** timer = setInterval(()=>{  
 counter++;  
 socket.send(**"I am Server - "**+counter);  
 },1000);  
 *//5:接收客户端发送消息 绑定事件message* socket.on(**"message"**,(msg)=>{  
 ***console***.log(**"服务器接收到消息"**+msg);  
 });  
 *//6:WS服务器接收客户端发来的断开连接请求* socket.on(**"close"**,()=>{  
 ***console***.log(**"客户端断开连接"**);  
 clearInterval(timer);  
 });  
})

* h5 WS协议客户端应用

<**h1**>WS客户端应用</**h1**>  
<**button id="tb1"**>连接ws服务器，并且接收消息</**button**>  
<**button id="tb2"**>向服务器发送一条消息</**button**>  
<**button id="tb3"**>断开到服务器的连接</**button**>  
<**script**>  
 **var *socket*** = **null**;  
 tb1.onclick = **function**(){  
 *//1:创建socket对象永久连接ws服务器* ***socket*** = **new** WebSocket(**"ws://127.0.0.1:9001"**);  
 *//2:接收服务器发来的消息* ***socket***.onmessage = **function**(e){  
 ***console***.log(**"接收服务器的一个消息"**+ e.data);  
 }  
 }*//17:40--17:45* tb2.onclick = **function**(){  
 *//向ws服务器发送一条消息* ***socket***.send(**"I am Client-"**+**new** Date().getTime());  
 }  
 tb3.onclick = **function**(){  
 *//断开ws服务器的连接* ***socket***.close();  
 }  
</**script**>