HTML5学习记录

一、Canvas

问题1：在canvas标签上设置了宽和高，与在css中设置的宽和高，显示的结果不一样，css设置的宽和高会将绘制的图进行拉伸。

解决1：canvas默认大小是：300px \* 150px ，相当于一张图片，css设置的宽高属性相当于对canvas这张图进行拉伸变化，所以绘制在canvas上的图也是相对css设置的宽高进行拉伸过的。所以要解决这个问题只有在canvas标签上直接设置宽高，替换默认宽高。

1、LineDashOffset属性：绘制虚线条的偏移量，初始值为0.0(float精度的数字)，值越小越往左。

setLineDash([x,y]):设置虚线条的属性值，x代表虚线的粗细，值越大越粗，y代表每个短条的间隔，值越大间隔越大。

2、可利用LineDashOffset属性编写跑马灯效果。其中lineDashOffset的值是负数时跑动的方向是顺时针，正数则是逆时针。利用setTimeout(方法名，延迟时间) 方法，实现跑动效果。（setTimeout(marquee,20)）

3、lineJoin属性：设置2个长度不为0的线条的相接处的链接样式。设置的值有3种：bevel（连接末端填充一个额外的以三角形为底的区域），round（连接部分以圆角展示，圆角的半径是线段的宽度），miter（默认值。连接部分填充一个菱形区域）。第一层是round，第二层是bevel，第三层是miter。

4、clip()方法：从canvas上裁剪部分区域，一旦剪切了某个区域，则之后所有的绘图都在剪切的区域内（不能访问画布除剪切区域外的地方）。若需要访问画布的其他区域，需要在使用clip()方法前调用save()方法保存对当前画布区域进行保存，并在以后的任意时间对其进行恢复（通过 restore() 方法）。

例：// Clip a rectangular area

ctx2.rect(50, 20, 200, 120); //从画布上绘制区域，该区域为clip()裁剪的区域

ctx2.stroke();

ctx2.save(); //调用save()方法，保存当前画布的操作

ctx2.clip(); //将其进行裁剪

// Draw red rectangle after clip() 在裁剪的区域进行其他操作

ctx2.fillStyle = "green";

ctx2.fillRect(0, 0, 150, 100); //在裁剪区域绘制矩形图片并填充颜色

ctx2.restore(); //调用restore()方法，重置回裁剪前的画布布局

ctx2.beginPath(); //调用beginPath()方法，开始在画布上绘制图形，包括被裁剪的区域。调用该方法可以防止绘制其他图片受影响

ctx2.rect(0, 0, 20, 50);

ctx2.fillStyle = "pink";

ctx2.fill();

ctx2.closePath();

5、渐变：

（1）线性渐变：createLinearGradient(渐变起点X坐标，渐变起点Y坐标，结点X坐标，结点Y坐标)；例：var gradient=ctx.createLinearGrident(20,20,40,50);

（2）径向渐变：createRadialGradient(渐变开始圆的X坐标，渐变开始圆的Y坐标，开始圆的半径，渐变结束圆的X坐标，渐变结束圆的Y坐标，结束圆的半径)；

（3）添加渐变色：addColorStop(位置，颜色值);位置的值介于0.0-1.0之间。

例：gradient.addColorStop(0,'#efefef'); gradient.addColorStop(1,'#b0b0b0');

（4）显示结果：fillStyle（填充）+fill()/fillRect()，strokeStyle（描边）+stroke()/strokeRect()，例：fillStyle='red'/'rgba(0,0,0,0.5)'/'rgb(234,234,234)'/添加渐变色的对象(gradient)。strokeStyle描边需要与lineWidth一起使用，lineWidth是设置线条的宽度。

6、文本：

（1）font：设置文本字体和字号。textAlign：水平对齐方式，textBaseLine：垂直对齐方式。

（2）显示：fillText(需显示的文字，相对于画布的X坐标，相对于画布的Y坐标，允许的最大文本宽度(可选))，填充文本，即实心文本。strokeText()参数与fillText一样，只是无填充，只是描边，即空心文本。

7、阴影：

shadowColor：阴影颜色；shadowBlur：阴影模糊级别，数值越大越模糊；shadowOffsetX/shadowOffsetY：阴影距形状的水平/垂直距离。

8、形状：

（1）矩形：rect(距左上角的X坐标，距左上角的Y坐标，宽度，高度)

（2）圆：arc(圆心X坐标，圆心Y坐标，半径，起始角(默认为0)，结束角(默认Math.PI\*2)，方向顺时针false/逆时针true);

（3）线条：

moveTo(X,Y),lineTo(X,Y)；

quadraticCurveTo(控制点X坐标，控制点Y坐标，结束点X坐标，结束点Y坐标)，绘制二次贝塞尔曲线需要确定路径是否存在，不存在则需要用beginPath()和moveTo()来定义起点；

bezierCurveTo(控制点1的X坐标，控制点1的Y坐标，控制点2的X坐标，控制点2的Y坐标，结束点X坐标，结束点Y坐标);三次贝塞尔曲线。

9、转换：

scale(x轴缩放值，y轴缩放值)，x轴代表宽度缩放（1=100%，0.5=50%），y轴代表高度缩放值。

rotate(旋转角度)，以弧度计算。如需将角度转换为弧度，使用公式：degree\*Math.PI/180，例：5度：5\*Math.PI/180。

translate(x,y)：重新映射画布上的(0,0)位置。

transform(水平缩放，水平倾斜，垂直倾斜，垂直缩放，水平移动，垂直移动)

替换当前的变换矩阵, transform() 允许您缩放、旋转、移动并倾斜当前的环境。

10、图像：

（1）drawImage()可在画布上绘制图像、画布或视频，也可绘制图像的某些部分，增加或减少图像的尺寸。

（a）定位：drawImage(img,x,y)，img=原图路径，x/y=定位的坐标

（b）定位图像并规定宽高：drawImage(img,x,y,w,h)

（c）剪切图像，并在画布上定位被剪切的部分：drawImage(img,sx,sy,sw,sh,x,y,w,h)，sx/sy=开始剪切的X/Y坐标，sw/sh=被剪切的图像宽/高，x/y=画布上放置图像的X/Y坐标，w/h=要使用的图像宽/高

二、CSS中的布局

CSS中添加了列布局，帮助分割文本变为列。

1、指定列数：column-count:3; //默认写法,添加CSS hask兼容不同浏览器。

-webkit-column-count:3 ; //兼容chrome浏览器

-moz-column-count:3 ; //FireFox浏览器

2、两列之间的间距：column-gap:10px; -webkit-column-gap:10px; -moz-column-gap:10px ;

3、两列之间的线条：column-rule:1px solid red;

三、Web Worker

1、Web Worker能帮助异步执行JavaScript文件

2、限制因素：a、不能访问/修改DOM节点；

b、不能访问全局变量或全局函数；

c、不能访问Window,document等一类的窗口属性。(alert,confirm)

但可以使用setTimeout(),setInterval()之类的函数，也可以使用XMLHttpRequest对象来做Ajax通信。

3、基本原理：在当前javascript的主线程中，使用Worker类加载一个js文件来开辟一个新线程，起到互不阻塞执行的效果，并且提供主线程和新线程交换数据的接口：postMessage，onMessage。结果从event.data中取得。

主线程代码：

//当主线程worker收到来自子线程的消息后，触发message事件

var worker = new Worker('../../js/worker.js');

worker.onmessage = function(event) {

document.getElementById('result').innerHTML+=event.data+"<br/>" ;

//主线程使用terminate方法中断与子线程来往，在浏览器中只能显示一次event.data

worker.terminate();

};

子线程代码：计算结果并将数据返回给主线程

//当主线程发来信息后，触发该message事件

var i = 0;

function timedCount() {

for (var j = 0, sum = 0; j < 100; j++) {

for (var i = 0; i < 100000000; i++) {

sum += i;

}

}

// 调用 postMessage 向主线程发送消息

postMessage(sum);

}

postMessage("Before computing," + new Date());

timedCount();

postMessage("After computing," + new Date());

或者 绑定onmessage方法接收主线程发送的数据

var fibonacci =function(n) {  
 return n <2? n : arguments.callee(n -1) + arguments.callee(n -2);  
};

onmessage =function(event) {  
 var n = parseInt(event.data, 10);  
 postMessage(fibonacci(n));  
};

四、本地存储（Local Storage）

1、数据添加采用键值对的方式，运用setItem(key,value)。

2、获取数据运用getItem(key);

3、存储Json格式可使用JSON.stringify();

4、清除键值对使用removeItem()，如果希望一次性清除所有的键值对，可使用clear()。

5、cookies和本地存储(Local Storage)的区别：

（1）cookies的数据客户端和服务端都能访问，cookie的数据通过每一个请求发送到服务端；Local Storage只能本地浏览器端访问数据，服务器不能访问本地存储，除非故意通过POST或GET的通道发送到服务器。

（2）每个cookie有4095byte，Local Storage每个域5MB

（3）cookies有有效期，过期后数据会被删除，Local Storage没有有效期。

6、事务存储（会话存储）：与本地存储类似，在数据仅在会话中有效，关闭浏览器时数据会被删除。

创建会话存储需使用“sessionStorage.variablename”,variablename是自定义变量名字。

例：创建一个统计访问该网页次数的会话存储

if(sessionStorage.visitCount){

sessionStorage.visitCount=Number(sessionStorage.visitCount)+1

}

else{

sessionStorage.visitCount=0;

}