## 基础步骤

1.设置画布

<canvas> </canvas>

默认没有颜色边框 默认大小300\*150

1.1设置大小：

<canvas width=”600” height=”400”> </canvas>

不能在style中设置画布大小 style中是缩放原始画布大小

1. 准备绘图工具

写在script中

2.1 获取画布

var canvas = document.querySelector('canvas');

2.2 获取绘图工具 获取绘图上下文

Var ctx=canvas.getContext(‘2d’);

2.3 设置起始点 终点 此时绘制的只是轨迹

Ctx.moveTo(100,100-0.5); 坐标相对于画布的左上角

Ctx.lineTo(200,100-0.5);

2.4 设置样式

ctx.linewidth=2; 线宽不用加px

Ctx.strokeStyle=’red’; 边框颜色

2.5 描边

ctx.stroke();

如果每条线样式不一样，则每条边都要描一次边

2.6 样式分隔线 开启新路径(限制后面的stroke的作用范围)

ctx.beginPath();

注：线条的默认颜色为黑色 默认宽度1px 线条中间对准刻度

处理线条发虚：在绘制时-/+ 0.5

## 设置样式

### 画笔的状态

lineWidth 线宽，默认1px

lineCap 线末端类型：(butt默认)、round圆、square方

lineJoin 相交线的拐点 miter(默认)、round圆、bevel平点

strokeStyle 线的颜色

fillStyle 填充颜色

getLineDash() 获取虚线宽度集合

lineDashOffset 设置虚线偏移量（负值向右偏移）

### 设置折点

Ctx.moveTo(100,100);

Ctx.lineTo(150,50); 折点

Ctx.lineTo(200,100);

lineJoin 相交线的拐点 miter(默认)、round、bevel

### 设置虚线

setLineDash([10,20,30]) 数组数字表示虚线的长短规律

10px实线 20px空白 30px实线 10px空白 ……

获取虚线规律：

Ctx.getLineDash();

虚线的偏移：

Ctx.lineDashOffset=5;

为正值时 整条虚线往左偏

其他API

Ctx.ClosePath(); 自动闭合路径

Ctx.stroke();描边

Ctx.strokeStyle=’red’; 线的颜色

Ctx.fill(); 自动填充

Ctx.fillstyle=’yellow’; 填充的颜色

## 非零环绕填充规则

向外引出一条线M

穿过M的线的方向 两边方向相同 则不填充 两边方向不相同 则填充

## 绘制点

设置点的位置

Var coordinate={

X:100,

Y:100

}

设置点的大小

Var pointSize=20;

绘制

ctx.moveTo(coordinate.x - pointSize / 2, coordinate.y - pointSize / 2);

ctx.lineTo(coordinate.x + pointSize / 2, coordinate.y - pointSize / 2);

ctx.lineTo(coordinate.x + pointSize / 2, coordinate.y + pointSize / 2);

ctx.lineTo(coordinate.x - pointSize / 2, coordinate.y + pointSize / 2);

ctx.closePath();

ctx.fill();

## 绘制矩形

Ctx.rect(a,b,c,d)； 绘制的是路径 需要去描边绘制填充

起始X坐标 起始Y坐标 矩形的宽度 矩形的高度

ctx.strokeRect(100,100,300,100); 已经描边的矩形

ctx.fillRect(100, 100, 300, 100); 已经填充的矩形

ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height);

清除 擦拭 某个矩形区域内的内容

## 渐变

### Css

linear-gradient (渐变方向 起始颜色 结束颜色)

渐变方向 to right to top to bottom to left

特殊的角度 使用deg 默认的角度是朝上 角度为正 顺时针转

例子：

background: linear-gradient(0deg,yellow,blue);

background: linear-gradient(to right, yellow 0% ,red 50% , blue 100%);

X%为每个颜色所占的比例

### Css3

fillStyle 可以使用颜色 也可以使用 渐变方案

var linearGradient = ctx.createLinearGradient(100,100,400,100);

设置渐变的： 起始的坐标 结束的坐标

linearGradient.addColorStop(0,'yellow'); 第一个数为每个颜色所占的比例 （0 ~ 1）

linearGradient.addColorStop(0.5,'red');

linearGradient.addColorStop(1,'blue');

ctx.fillStyle = linearGradient;

ctx.fillRect(100, 100, 300, 100); 已填充的矩形

## 绘制圆

绘制圆弧的方向 默认是顺时针方向false 逆时针就是true

ctx.arc(x0 , y0 , 150 , 0 , 2 \* Math.PI ，true );

圆心 圆的半径 起始弧度 结束弧度 方向

一个圆的完整弧度是 2π

别忘了写ctx.stroke();

## 文字

### 显示文字

Var str=”我最美”;

ctx.strokeText(str,x0,y0); 描边的文字 文字 X坐标 Y坐标

ctx.fillText(str,x0,y0); 填充的文字

文字的定位是基于文字的左下角

Ctx.font=’40px 宋体’; 设置字体

### 测量文字的长度

ctx.measureText( str ); 得到的是对象

ctx.measureText( str ).width; 取长度

### 文字对齐

这里的对齐是基于坐标点的对齐

Ctx.textAlign=’center’; 左右居中

Center right left right 则文字的最右在正中间

Ctx.textBaseline=’middle’; 上下居中

Top bottom middle

## 绘制图片

必须先绑定事件 再加载图片 防止加载过快不执行load

Image.onload=function(){

// 图片加载完成事件

}

Image.src=’images/01.jpg’;

### 将图片截到画布上

要写在图片加载事件中

用法一：

Ctx.drawImage( image ,100 ,100 ,100 ,100 );

图片元素 \ canvas中的X坐标 \ Y坐标 \ 图片显示大小 宽度 \ 高度

用法二：

Ctx.drawImage( image ,100 ,100 ,100 ,100 ,100 ,100 ,100 ,100 ); 不能写百分比

图片中的X坐标 Y坐标（截图开始的位置）

图片裁剪的区域

图片在canvas中的定位（从画布的哪个点开始绘制图片）

图片在canvas中的显示大小 宽度 \ 高度