01

canvas：

- 就是HTML5的一个标签，提供一个空白的图形区域，起到画布的作用，用来展示，本身不具有绘图功能
- 需要通过**js中的绘制API**来绘制

```
1 通过js中提供的绘图API来进行图形绘制
2 canvas能够将绘制的内容展示到浏览器中给用户查看
```

OpenGL专门处理3D图形

webgl浏览器端处理3D效果的

区分rect和fillRect

```
/*ctx.fillStyle = 'red';
ctx.fillRect(100, 100, 200, 160)

ctx.fillStyle = 'green';
ctx.fillRect(150, 150, 200, 160)*/

ctx.fillStyle = 'red';
ctx.rect(100, 100, 200, 160);
ctx.fill();

ctx.fillStyle = 'green';
ctx.rect(150, 150, 200, 160);
ctx.fill();
```

fillRect不会影响之前的红色，rect会影响

```
// canvas中绘图是基于状态的！
// 路径 和 状态 是完全不同的内容，不会相互影响！
// 开启新的路径，不会影响到状态！
```

```
2 路径 和 状态的区别：
路径：  moveTo / lineTo 绘制路径，stroke/fill 绘制路径（描边/填充）
       beginPath 开启新路径

状态：  strokeStyle / fillStyle
```

---

canvas逐渐取代Flash，比Flash更加立体，更加精巧，**canvas游戏在流畅度和跨平台方面更牛**

可视化数据（数据图表化）： 百度的 echarts或者

3D动态效果： 可以通过 three.js官网

canvas主要应用的领域（了解）

- 1、游戏：canvas在基于Web的图像显示方面比Flash更加立体、更加精巧，canvas游戏在流畅度和跨平台方面更牛。
- 2、**可视化数据**（数据图表话），比如: 百度的ECharts、d3.js、three.js highcharts
- 3、**banner广告**：Flash曾经辉煌的时代，智能手机还未曾出现。现在以及未来的智能机时代，HTML5技术能够在banner广告上发挥巨大作用，用Canvas实现动态的广告效果再合适不过。
- 4、未来
  - 模拟器：无论从视觉效果还是核心功能方面来说，模拟器产品可以完全由JavaScript来实现。
  - 远程计算机控制：Canvas可以让开发者更好地实现基于Web的数据传输，构建一个完美的可视化控制界面。
  - 图形编辑器：Photoshop图形编辑器将能够100%基于Web实现。

---

canvas默认尺寸为：300（宽）* 150（高）

```
<canvas>
  canvas 默认的尺寸为：  300（宽） * 150（高）

  只有浏览器不兼容canvas的时候，canvas中的内容才会展示出来！

  您的浏览器不支持canvas，请更新您的浏览器 <a href="http://web.itcast.cn/">
  传智博客-前端</a>
</canvas>
```

---

canvas使用步骤：

1. 获取到canvas标签
2. 获取到 当前canvas标签的绘制上下文
3. 调用 上下文（对象）中的绘图API进行绘制

```
var cv = document.getElementById('cv');
// context 上下文
// CanvasRenderingContext2D 就是构造函数，ctx 其实就是一个实例对象
// getContext可以被看作成一个 工厂函数
var ctx = cv.getContext('2d');
// console.log( ctx );
```

？？？？ DOM.$('')

方法：

- ctx.moveTo(x,y) ：将画笔移到起始点
- ctx.lineTo(x,y) ： 描绘路径，如果没有moveTo，第一个lineTo就相当于moveTo

```
// 绘制一条线段
// moveTo 将canvas的画笔移动到某个起始点
// 第一个参数： 表示 x 坐标
// 第二个参数： 表示 y 坐标
ctx.moveTo(100, 100);

// 描绘路径，
// 第一个参数：表示x坐标
// 第二个参数：表示y坐标
ctx.lineTo(300, 100);

// 将已经描绘的路径绘制到画布中
// 描边
ctx.stroke();
```

- ctx.stroke() ：将已经描绘的路径 绘到画布中
- ctx.fill() ：填充，如果描绘的路径没有闭合，会自动闭合，再填充
- ctx.beginPath() ：**开启新路径**，相当于将之前绘制上下文中描绘的路径清除掉
  - **调用该方法后，再 调用fill或stroke方法，就不会描绘以前的路径了**，只描绘新路径的
  - **路径的操作不会影响绘制状态（strokeStyle或fillStyle），也就是beginPath之前绘制的状态，后面再 fill/stroke 的时候也会拥有此状态**
- ctx.closePath() ：闭合路径,比如画正方形，只画3条边，然后调用ctx.closePath()会自动连最后一条闭合

```
// 闭合路径：仅仅是在当前路径的起始位置与结束位置之间做一条连线
ctx.moveTo(100, 100)
ctx.lineTo(200, 100)
ctx.lineTo(200, 200)
ctx.lineTo(100, 200)
// ctx.lineTo(100, 100)

ctx.closePath();

ctx.stroke();
```

- ctx.rect（x,y,w,h）： x坐标，y坐标，宽，高
  - 绘制矩形的路径 ，还需要配合stroke或者fill

```
// ctx.rect(x, y, w, h)

// 作用：描绘路径
ctx.rect(100, 100, 100, 100);

// ctx.stroke();
ctx.fill();
```

- ctx.strokeRect（x,y,w,h）：
- ctx.fillRect（x,y,w,h）：

```
// 快速绘制矩形：
// 注意：这两个方法不会产生路径，直接就是绘制矩形！！！
// ctx.strokeRect(x, y, w, h)
// ctx.fillRect(x, y, w, h)

// ctx.strokeRect(300, 100, 200, 200)
ctx.fillRect(300, 100, 200, 200)
```

不产生路径，直接绘制

- ctx.clearRect（x,y,w,h）：canvas的API只提供了 一个 清除绘制内容的方法
  - 作用：清除某个矩形区域里的内容，仅仅是擦除展示的内容，不会影响路径

```
// canvas画图的方式：
// 绘制的时候是将整个画布的内容绘制出来，擦出的时候也是将整个画布的内容全部
// 擦出，然后，再重新进行绘制！

// 如何清空整个画布？？？
ctx.clearRect(0, 0, cv.width, cv.height);
```

- ctx.clearRect（x,y,w,h）

属性：

ctx.strokeStyle：

```
设置描边样式
用来设置当前绘制图形的颜色
ctx.strokeStyle = 'green';
ctx.strokeStyle = '#f40000';
ctx.strokeStyle = 'rgb(255, 100, 100)';
ctx.strokeStyle = 'rgba(255, 0, 0, 0.3)';
ctx.stroke();
```

ctx.fillStyle：

```
// 设置填充样式
ctx.fillStyle = '
ctx.fill();
```

颜色同上

注意：一定先写 fillStyle= 之后再写 fill（）

设置canvas宽高：

- 第一种

```
<!-- 第一种方式： -->
<canvas width="600" height="400"></canvas>
```

- 第二种

```
var cv = document.getElementById('cv');
cv.height = 400;
cv.width = 600;
```

```
通过以上两种标准的设置宽度和高度，是改变了当前canvas画布中像素点的个数
实际canvas中的高度和宽度，分别表示：当前画布中垂直方向 和 水平方向
个数。通过标准设置宽高的方式，是改变了画布中像素点的个数。

但是通过样式来设置高度和宽度，实际上没有改变像素点的个数
```

注意：不能通过css设置宽高，因为通过css设置的宽高，相当于把canvas拉伸了，绘制图形时 是变形的

---

canvas坐标：

- 左上角是原点
- 向右　X正方向
- 向下　Y正方向

---

canvas中的上下文（说白了就是对象）：

- 相当于画图软件的工具栏，里面提供了很多与绘图相关的工具

```
本质上 绘制上下文 就是一个对象，对象中有属性和方法
```

1. 获取绘图平面图形的上下文
    a. **var cv = document.getElementById('dv');**
2. ctx就是绘图上下文，里面包含与绘图相关的各种API（颜色，绘制线等方法）
    a. **var ctx = cv.getContext('2d');**

```
ctx 就是绘图上下文

属性描述的是：绘制状态
方法用来绘制内容
```

```
// 获取绘制 3D 图形的上下文
var cv = document.createElement("canvas");
// OpenGL
console.log(cv.getContext("webgl"));
```

```
ctx.moveTo(100, 100)
ctx.lineTo(200, 100)
ctx.lineTo(200, 200)
ctx.lineTo(100, 200)
ctx.lineTo(100, 100)

// 将已经描绘好的所有路径全部绘制到当前画布中！！！
ctx.stroke();
```

画多个图形，也最后只写一个stroke，此方法绘制所有路径，多写会重复，线会变粗

stroke（）：

```
将已经描绘好的所有路径全部绘制到当前画布中！！！
最佳实践：先将所有的路径描绘好，然后一次调用 stroke 进行所有路径
的绘制！！！否则，会造成重复绘制的问题！！！
```

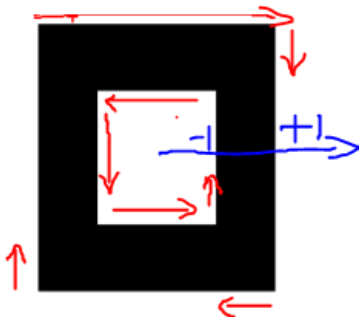
---

ctx.beginPath（）

```
ctx.moveTo(100, 100)
ctx.lineTo(200, 100)
ctx.strokeStyle = 'red';
ctx.stroke();

// 开启新路径，只绘制以后再描绘的路径
ctx.beginPath();

ctx.moveTo(200, 100)
ctx.lineTo(200, 200)

ctx.strokeStyle = 'green';
ctx.stroke();
```

非零环绕原则：用来查看某个区域是否被填充的一个规则

- 从要查看的区域引出一条射线，穿过整个图形，使射线与图形相交，并且以射线起点为圆心，顺着线绘制方向画圆，**顺时针**形成的圆记为**+1**，**逆时针-1**，所有结果相加，结果为0，射线原点的区域不绘制，**结果不为0，绘制**



边擦边画，运动的矩形（不会很影响性能，比如做游戏也相同道理**）**

```
// 绘制运动的矩形：
var startX = 100,
  stopX = 400,
  step = 3;

var timerId = setInterval(function() {
  if( startX >= stopX ) {
    startX = stopX;
    clearInterval( timerId );
  }

  // 清空画布
  ctx.clearRect(0, 0, cv.width, cv.height);
  // 绘制矩形
  ctx.fillRect(startX, 100, 100, 100);

  startX += step;
}, 50);
```