爱创课堂前端培训

# canvas

第1天课堂笔记（本课程共1天）

班级：北京前端训练营19期

讲师：彭帅伟

日期：2019年1月4日

彭帅伟老师

QQ: 284337908

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

目录

[canvas 1](#_Toc21243)

[复习： 3](#_Toc562)

[一、 canvas 5](#_Toc598)

[1.1坐标系 7](#_Toc5500)

[1.2 API 8](#_Toc10890)

[二、 绘制图片 10](#_Toc9301)

[下午复习： 11](#_Toc24261)

[三、 状态的保存与恢复 13](#_Toc32752)

[四、 获取canvas上的像素 13](#_Toc4906)

[4.1 getImageData 13](#_Toc31930)

[4.2 putImageData 14](#_Toc0)

[五、 融合 14](#_Toc31040)

[5.1 ctx.globalCompositeOperation 14](#_Toc31604)

# 复习：

HTML5:

本地存储：

用于存储数据， 是一个对象

对象名称： localStorage

它的生命周期是永久的

API:

存储数据：

localStorage.setItem(key, value)

获取数据：

localStorage.getItem(key)

删除某一项：

localStorage.removeItem(key)

删除全部：

localStorage.clear()

会话存储：

对象名称： sessionStorage

它的生命周期是从浏览器打开到关闭

多线程：

js是单线程，当js执行的时候， 页面是停止渲染的， 只有当js执行完毕之后，页面才可以渲染出来

HTML5提供了一个Worker函数， 用于开辟一个新的线程

使用方式：

1 将执行的代码提取到一个js文件中

2 初始化Worker函数

特点：需要服务器环境

传递信息:

开辟的额外的线程可以通过postMessage将结果发送主线程

主线程可以通过onmessage来接受另外的一个线程推送的信息

WebSql:

openDatabase(dbname, version, descript, size)

作用就是用来打开或者是创建数据库

创建数据库之后返回一个数据库对象

通过该对象调用transaction，并传递一个回调函数， 函数中的参数就是事物对象

通过调用 该事物对象中原型上的executeSql执行sql语句

使用方式：

ta.executeSql(sql, [], sucessCallback, failedCallback)

EventSource:

用于主动向浏览器端推送信息

使用方式与Worker函数类似， 都要进行初始化

history:

history.forward:

该方法会加载历史记录列表中的下一个URL

调用该方法等价于点击了前进按钮或者是history.go(1)

history.back:

该方法会加载历史记录列表中的上一个URL

调用该方法等价于点击了后退按钮或者是histroy.go(-1)

pushState(obj, title, url):

添加新的历史记录

replaceState(obj, title, url):

替换当前的历史记录

拖拽事件：

ondrag: 拖拽

ondragstart: 拖拽开始

ondragend: 拖拽结束

ondragenter: 拖拽进入

ondragleave: 拖拽离开

ondragover： 悬浮

ondrop: 丢弃， 该事件有一个bug，悬浮事件中的默认行为导致该事件不能执行，如果想要添加ondrop事件， 必须给该元素添加ondragover并且阻止默认事件

# canvas

HTML5新增了canvas标签，用于提供“画布”

画笔要通过js来获取

canvas的标准属性有width和height（例如： id、class都是通用标准属性）

width: 表示canvas的宽

height: 表示canvas的高

举例：

|  |
| --- |
| 1. <!-- canvas是HTML5提供的一个新的标签， 用于提供“画布” --> 2. <canvas id="myCanvas" width="800" height="400"></canvas> |



获取画笔的方式：

1 获取相应的canvas元素

|  |
| --- |
| 1. // 获取相应的canvas元素 2. var canvas = document.getElementById("myCanvas"); |

2 通过canvas获取画笔

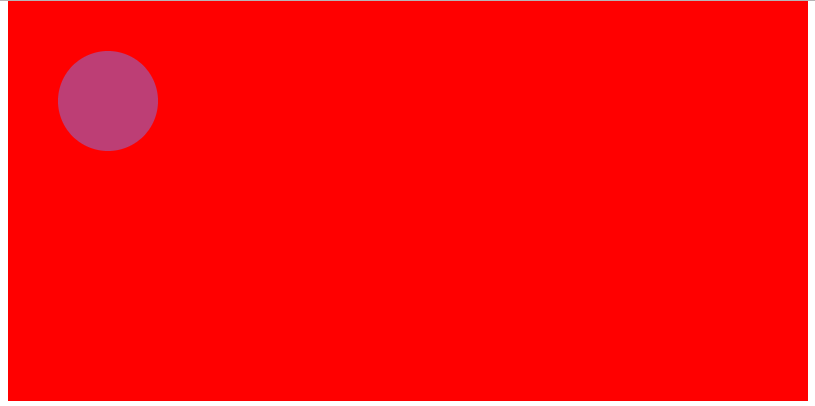
|  |
| --- |
| 1. // 通过canvas元素来获取画笔 2. var ctx = canvas.getContext("2d"); |

查看画笔：

|  |
| --- |
| 1. 属性：  1. 方法： |

绘制圆：

|  |
| --- |
| 1. // 改变填充色 2. ctx.fillStyle = "rgba(123, 123, 234, .5)"; 3. // 画圆 4. // 开启路径 5. ctx.beginPath(); 6. // 绘制弧 7. // ctx.arc(100, 100, 50, 0, Math.PI \* 2); 8. // 关闭路径 9. ctx.closePath(); 10. // 填充 11. ctx.fill(); |



## 1.1坐标系

canvas已经是最底层元素了， 因此定位坐标系对于canvas是没有意义的， 但是canvas是用来显示图像的， 所以自带了一个坐标系， 默认是与元素的定位坐标系是一致

举例：

|  |
| --- |
| 1. // canvas是一块区域，用于显示图像的， 所以自带了一个坐标系 2. // x轴正方向向右， y轴正方向向下 3. // 坐标轴位于左上角，与元素的定位坐标系是一致的 4. // 获取元素 5. var canvas = document.getElementById("myCanvas"); 6. // 获取画笔 7. var ctx = canvas.getContext("2d"); 8. ctx.font = "20px 宋体"; 9. // X轴 10. line(5, 5, 790, 5); 11. line(785, 0, 790, 5); 12. line(785, 10, 790, 5); 13. // Y轴 14. line(5, 5, 5, 590); 15. line(0, 585, 5, 590); 16. line(10, 585, 5, 590); 17. // 绘制文字 18. ctx.fillText("x轴", 750, 40); 19. ctx.fillText("y轴", 30, 580); 20. // 封装一个函数，传递两个点， 绘制两个点之间的线段 21. function line(x1 ,y1, x2, y2) { 22. // 开启路径 23. ctx.beginPath(); 24. // 移动画笔到某个位置 25. ctx.moveTo(x1, y1); 26. // 移动画笔到x2, y2的位置 27. ctx.lineTo(x2, y2); 28. // 描边 29. ctx.stroke(); 30. } |

结果：

|  |
| --- |
|  |

## 1.2 API

canvas上是用来绘制图形的，大部分都是在操作路径

所以要开启路径

beginPath()

closePath()： 闭合路径 在关闭路径时候的一个点，与开始时候的一个点之间会形成一条线段

fillRect(x, y, w, h)

作用: 填充矩形

x: 在canvas上的x点

y：在canvas上的y点

w: 矩形的宽

h: 矩形的高

strokeRect(x, y, w, )

作用： 描边矩形

x: 在canvas上的x点

y: 在canvas上的y点

w: 矩形的宽

h: 矩形的高

clearRect(x, y, w, h)

作用: 清除canvas上的一块区域

x: canvas上的x点

y: canvas上的y点

w：要清除的区域的宽

h: 要清除的区域的高  
  
arc(x, y, r, star, end, bool)

x: 圆弧所在圆心的x点

y: 圆弧所在的圆心的y点

star: 圆弧的起始位置

end: 圆弧的终点位置

bool: 方向 false: 顺时针 true: 逆时针

fill: 填充

stroke: 描边

fillStyle: 改变填充色

strokeStyle: 改变描边色

font: 改变字体

lineWidth: 改变线宽

fillText: 绘制文字

# 绘制图片

在canvas上绘制图片有三种方式：1 插入图片 2 设置背景图 3 通过drawImage

drawImage有三种使用方式：

第一种使用方式：

drawImage(img, x, y)

img: 要绘制的图片

x: 将原尺寸的图片放在canvas上的x点

y: 将原尺寸的图片放在canvas上的y点

第二种使用方式：

缩放图片

drawImage(img, x, y, w, h)

img: 要绘制的图片

x: 将缩放后的图片放在canvas上x点

y: 将缩放后的图片放在canvas上的y点

w: 缩放后的图片的宽

h: 缩放后的图片的高

第三种使用方式：

截取图片中的某一部分

drawImage(img, img\_x, img\_y, img\_w, img\_h, canvas\_x, canvas\_y, canvas\_w,canvas\_h)

img: 要绘制的图片

img\_x: 要截取的图片的x点

img\_y: 要截取的图片的y点

img\_w: 要截取的图片的宽

img\_h: 要截取的图片的高

canvas\_x: 将截取后的图片放在canvas上的x点

canvas\_y: 将截取后的图片放在canvas上的y点

canvas\_w: 将缩放后的图片放在canvas上的宽

canvas\_h: 将缩放后的图片放在canvas上的高

## 下午复习：

canvas: 是HTML5新增的一个标签，用于提供“画布”

获取画笔要通过canvas元素来获取

坐标系：

canvas是用来显示图形的， 所以自带了一个坐标系， 默认是与元素的定位坐标系是一致的

API:

ctx.arc(x, y, r, star, end, bool)

ctx.fillRect(x, y, w, h)

ctx.strokeRect(x, y, w, h)

ctx.clearRect(x, y, w, h)

ctx.fill(): 填充

ctx.stroke(): 描边

ctx.fillStyle: 改变填充色

ctx.strokeStyle: 改变描边色

ctx.font: 设置文字

ctx.lineWidth: 改变线宽

绘制图形：

通过drawImage绘制图片

有三种使用方式：

1 以原尺寸绘制图片

ctx.drawImage(img, x, y)

img: 绘制的图片

x：以原尺寸将图片放在canvas上的x点

y: 以原尺寸将图片放在canvas上的y点

2 缩放图片

ctx.drawImage(img, x, y, w, h)

img: 要绘制的图片

x: 将缩放后的图片放在canvas上的x点

y: 将缩放后的图片放在canvas上的y点

w: 缩放后的图片的宽

h：缩放后的图片的高

3截取图片的某一部分

ctx.drawImage(img, img\_x, img\_y, img\_w, img\_h, canvas\_x, canvas\_y, canvas\_w, canvas\_h)

img：绘制的图片

img\_x, img\_y, img\_w, img\_h: 这四个参数共同决定要截取的图片的区域

canvas\_x, canvas\_y, canvas\_w, canvas\_h: 这四个参数共同决定将截取后的图片放在canvas上的位置

# 状态的保存与恢复

在canvas上操作图像的时候很有可能用到之前的状态，canvas提供了相应的api用来保存状态

举例：

|  |
| --- |
| 1. // 获取元素 2. var canvas = document.getElementById("myCanvas"); 3. // 获取画笔 4. var ctx = canvas.getContext("2d"); 5. // 改变填充色 6. ctx.fillStyle = "orange"; 7. // 绘制矩形 8. ctx.fillRect(0, 0, 100, 100); 9. // 保存状态 10. ctx.save(); 11. // 改变填充色 12. ctx.fillStyle = "blue"; 13. // 绘制矩形 14. ctx.fillRect(100, 100, 100, 100); 15. // 恢复状态 16. ctx.restore(); 17. // 绘制矩形 18. ctx.fillRect(200, 200, 100, 100); |

save方法可以使用多次，每save一次就把当前canvas上的状态保存一份，每restore一次就用掉了一次canvas上的状态

每save一次就相当于往子弹夹中安装了一颗子弹

每restore一次就相当于扣了一次扳机

# 获取canvas上的像素

## getImageData

通过该方法可以获取canvas上的像素信息， 返回的是一个对象  
 使用方式：

ctx.getImageData(x, y, w, h)

x: 要获取的矩形区域的x点

y: 要获取的矩形区域的y点

w：矩形的宽

h: 矩形的高

举例：

|  |
| --- |
| 1. // 获取像素信息 2. getPX.onclick = function() { 3. var imgData = ctx.getImageData(0, 0, 1024, 680); 4. console.log(imgData); 5. } |

查看像素对象：

|  |
| --- |
|  |

## putImageData

该方法是将修正之后的像素对象重新放回到canvas中

使用方式：

ctx.putImageData(imgData, x, y)

imgData: 修正之后的像素对象

x: 放在canvas上的x点

y: 放在canvas上的y点

举例：

|  |
| --- |
| 1. // 去掉红色 2. red.onclick = function() { 3. var imgData = ctx.getImageData(0, 0, 1024, 680); 4. // 循环将红色去掉 5. for (var i = 0; i < imgData.data.length; i += 4) { 6. imgData.data[i] = 0; 7. } 8. // 将修正之后的像素对象放回canvas上 9. ctx.putImageData(imgData, 0, 0); 10. } |

# 融合

所谓融合， 就是在canvas上显示图形的时候，新图形与原有图形之间的覆盖关系

默认情况下， 新的图形覆盖原有图形

## 5.1 ctx.globalCompositeOperation

source-over 默认。在目标图像上显示源图像。

source-atop 在目标图像顶部显示源图像。源图像位于目标图像之外的部分是不可见的。

source-in 在目标图像中显示源图像。只有目标图像内的源图像部分会显示，目标图像是透明的。

source-out 在目标图像之外显示源图像。只会显示目标图像之外源图像部分，目标图像是透明的。

destination-over 在源图像上方显示目标图像。

destination-atop 在源图像顶部显示目标图像。源图像之外的目标图像部分不会被显示。

destination-in 在源图像中显示目标图像。只有源图像内的目标图像部分会被显示，源图像是透明的。

destination-out 在源图像外显示目标图像。只有源图像外的目标图像部分会被显示，源图像是透明的。

lighter 显示源图像 + 目标图像。

copy 显示源图像。忽略目标图像。

xor 使用异或操作对源图像与目标图像进行组合。

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |