EXERCISE

（JavaScript Canvas DAY 02）

版本编号 1.0

JavaScript Canvas DAY02

1. 使用Canvas技术制作刮刮卡效果，当

运行结果如图1



图- 1

* 参考答案

此案例需要用户创建html文件，进行网页布局，创建Canvas标签，使用js进行canvas的操作，清除canvas的颜色，来达到刮刮卡的效果。

实现此案例需要按照如下步骤进行。

步骤一：新建HTML文件

首先新建一个名为index的html文件，进行网页布局，代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>刮刮卡</title>

</head>

<body>

<div id="wrap">

<img src="img/cark.png" alt="" />

<!--奖品-->

<p id="prize">5,000,000</p>

<!--行内元素-->

<canvas id="my-canvas" width="255" height="125">

该浏览器不支持canvas元素

</canvas>

</div>

</body>

</html>

步骤二：进行网页修饰

对照效果图，使用css进行网页的修饰，代码如下所示：

\* {

padding: 0;

margin: 0;

}

#wrap {

width: 320px;

height: 568px;

margin: 0 auto;

position: relative;

}

#wrap>img {

width: 100%;

}

#my-canvas {

position: absolute;

top: 195px;

left: 32px;

transition: all 1s;

}

#prize {

width: 255px;

height: 125px;

position: absolute;

top: 195px;

left: 32px;

text-align: center;

line-height: 125px;

font-size: 40px;

color: gold;

}

步骤三：获取canvas的画笔

创建script标签，获取canvas元素，并获取画笔，代码如下所示：

//获取元素

var myCanvas = document.getElementById("my-canvas");

//设置宽和高

myCanvas.width = 600;

myCanvas.height = 600;

//上下文, canvas的画笔

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

步骤四：绘制矩形

在奖励区域，绘制矩形。代码如下所示：

//绘制矩形

context.fillStyle = "darkgray";//设置填充色

context.fillRect(0, 0, myCanvas.width, myCanvas.height);

步骤五：添加刮刮卡效果

添加鼠标点击事件和移动事件，来获取鼠标的位置，并把对应位置上的内容消除。代码如下所示：

//鼠标按下

myCanvas.onmousedown = function () {

//鼠标移动

myCanvas.onmousemove = function (event) {

//获取鼠标位置(在canvas坐标系下)

var x = event.clientX - myCanvas.offsetLeft - wrapDiv.offsetLeft;

var y = event.clientY - myCanvas.offsetTop - wrapDiv.offsetTop;

//设置图像合并方式

context.globalCompositeOperation = "destination-out";

//画圆

context.beginPath();

context.arc(x, y, 10, 0, Math.PI \* 2, true);

context.fillStyle = "cyan";

context.fill();

justify();

}

}

当鼠标松开时，需要将移动事件清除，否则会出现不用点鼠标就能清除的bug。代码如下所示：

//鼠标松开

myCanvas.onmouseup = function () {

//清除移动事件

myCanvas.onmousemove = null;

}

步骤六：优化刮刮卡效果

当清除的区域超过80%时，基本上已经可以看到奖励，没有必要让用户把所有的都刮完。所以每次清除某些内容，就需要进行判断，超过80%，其他部分自动清除。代码如下所示：

//判断是否清除80%

function justify () {

//获取图像数据

var imageData = context.getImageData(0, 0, myCanvas.width, myCanvas.height);

//获取颜色数据

var colorData = imageData.data;

//计算透明的像素点

var count = 0;

for (var i = 0; i < colorData.length; i += 4) {

//获取alpha值

if (colorData[i + 3] == 0) {

count++;

}

}

//判断

if (count > colorData.length / 4 \* 0.8) {

myCanvas.style.opacity = 0;

}

}

然后运行HTML文件，按下鼠标，并进行滑动即可进行刮刮卡。

本案例的完整代码如下所示：

代码量较大，请查看02\_练习源码/1刮刮卡

1. 制作房产证：根据输入框输入的名字，在房产证的户主位置上，填写上名字，并可以将制作的房产证图片导出

运行结果如图2

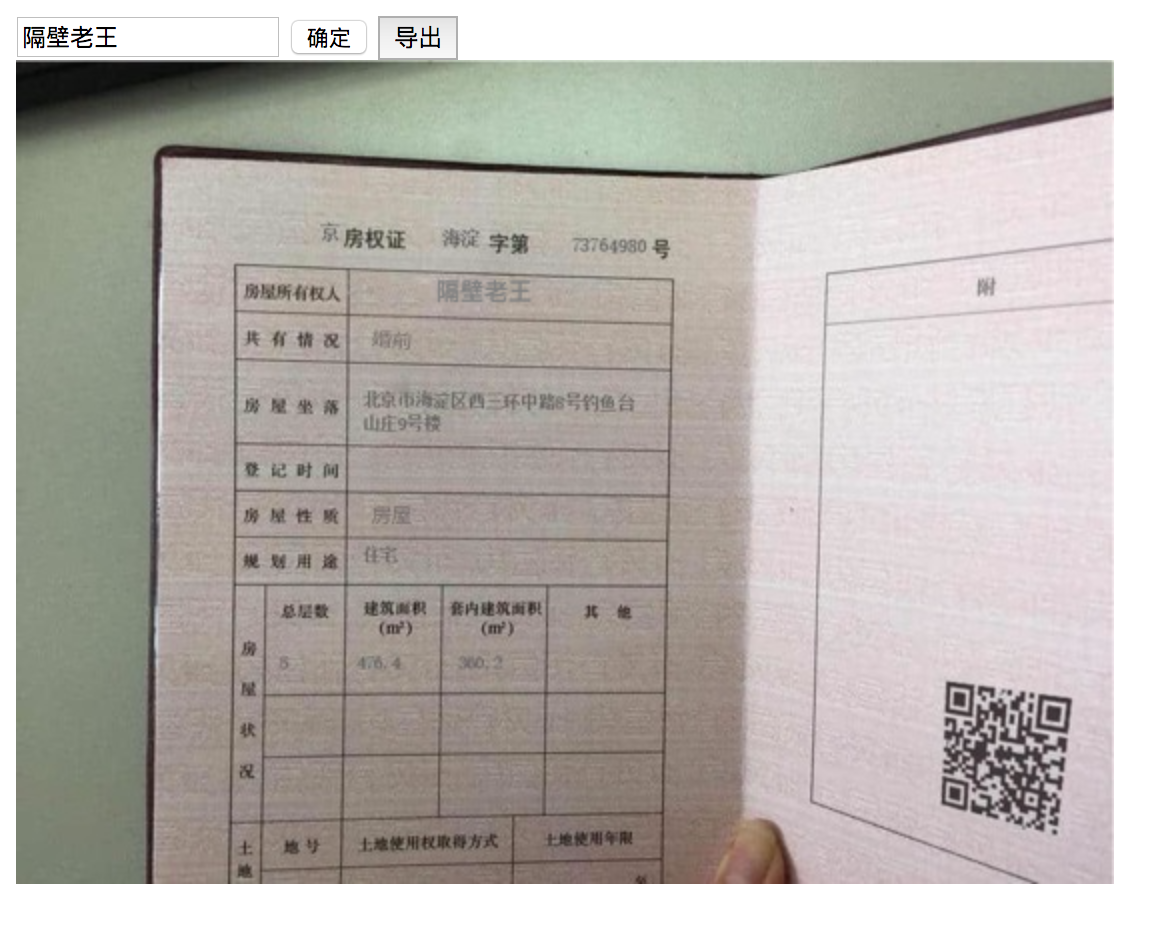


图- 2

* 参考答案

此案例需要用户创建html文件, 创建canvas标签，再通过js获取画笔，将图片绘制到画板上，并在指定的位置绘制文字。

实现此案例需要按照如下步骤进行。

步骤一：新建HTML文件，进行网页布局

首先新建一个名为index的html文件，创建canvas标签，代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<input type="text" placeholder="请输入房主姓名" />

<input type="button" value="确定" />

<button>导出</button><br />

<canvas id="my-canvas" width="549" height="412"></canvas>

</body>

</html>

步骤二：获取画笔

创建script标签，获取canvas元素，并获取绘制画笔。代码如下所示：

//获取canvas

var myCanvas = document.getElementById("myCanvas");

var ctx = myCanvas.getContext('2d');

步骤三：绘制图片

将图片绘制在画板上，需要注意的是：需要等待图片加载完成才能绘制，否则绘制失败。代码如下所示：

//图片元素

var img = new Image();

img.src = "img/zheng.jpg";

img.onload = function() {

//绘制图片

ctx.drawImage(img, 0, 0);

}

步骤四：绘制名字

为确认按钮添加点击事件，获取输入框的内容，并在图片上进行绘制。代码如下所示：

//确定按钮事件

var submitBtn = document.querySelector("input[type=button]");

submitBtn.onclick = function() {

//清除canvas内容

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

//图片

ctx.drawImage(img, 0, 0);

//文字

ctx.fillStyle = "gray";

var nameInput = document.querySelector("input[type=text]");

ctx.fillText(nameInput.value, 210, 120);

}

步骤五：导出图片

为导出按钮添加点击事件，获取对画板的内容进行base64编码，生成路径，在新的窗口显示。代码如下所示：

//导出按钮事件

var exportBtn = document.querySelector("button");

exportBtn.onclick = function() {

//导出canvas上的图片, 实质: 将图片的数据存到url中

var imgURL = myCanvas.toDataURL();

//console.log(imgURL);

//打开新的窗口

window.open(imgURL);

}

然后运行HTML文件，在输入框里输入名字，并点击确认按钮，即可看到效果。

本案例的完整代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<input type="text" placeholder="请输入房主姓名" />

<input type="button" value="确定" />

<button>导出</button><br />

<canvas id="my-canvas" width="549" height="412"></canvas>

<script type="text/javascript">

//获取元素

var myCanvas = document.getElementById("my-canvas");

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

//图片元素

var img = new Image();

img.src = "img/zheng.jpg";

img.onload = function() {

//绘制图片

ctx.drawImage(img, 0, 0);

}

//确定按钮事件

var submitBtn = document.querySelector("input[type=button]");

submitBtn.onclick = function() {

//清除canvas内容

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

//图片

ctx.drawImage(img, 0, 0);

//文字

ctx.fillStyle = "gray";

var nameInput = document.querySelector("input[type=text]");

ctx.fillText(nameInput.value, 210, 120);

}

//导出按钮事件

var exportBtn = document.querySelector("button");

exportBtn.onclick = function() {

//导出canvas上的图片, 实质: 将图片的数据存到url中

var imgURL = myCanvas.toDataURL();

//console.log(imgURL);

//打开新的窗口

window.open(imgURL);

}

</script>

</body>

</html>

1. 放大镜：使用canvas技术，实现放大镜效果

运行结果如图3

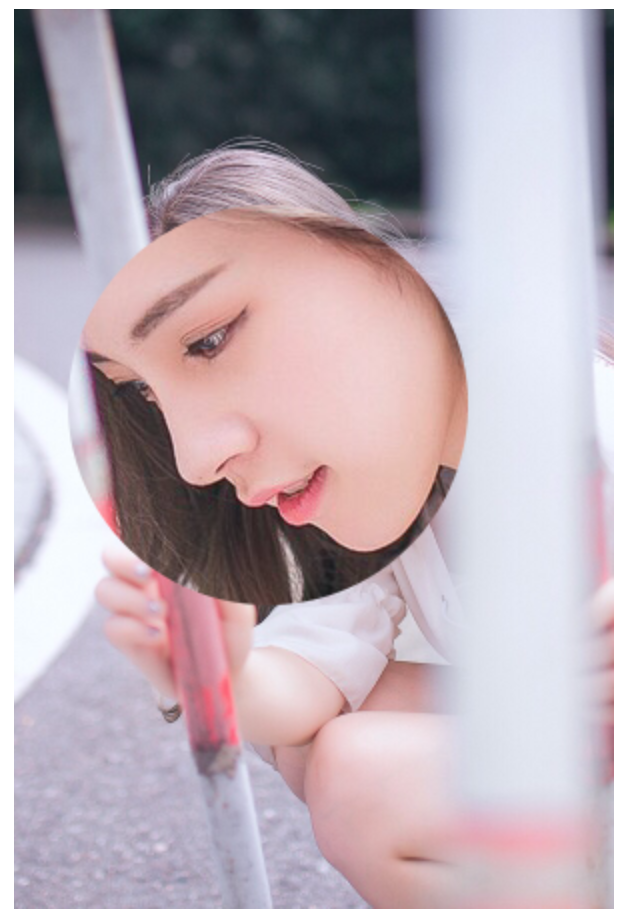


图- 3

* 参考答案

此案例需要用户创建html文件, 创建canvas标签，将图片绘制在画板中，通过鼠标获取指定范围内容的图片，进行放大。

实现此案例需要按照如下步骤进行。

步骤一：新建HTML文件，进行网页布局

首先新建一个名为index的html文件，创建canvas标签，代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#my-canvas {

display: block;

margin: 0 auto;

}

</style>

</head>

<body>

<canvas id="my-canvas" width="300" height="450"></canvas>

</body>

</html>

步骤二：获取画笔

创建script标签，获取canvas元素，并获取绘制画笔。代码如下所示：

//获取canvas

var myCanvas = document.getElementById("myCanvas");

var ctx = myCanvas.getContext('2d');

步骤三：绘制图片

将图片绘制在画板上，需要注意的是：需要等待图片加载完成才能绘制，否则绘制失败。代码如下所示：

//画图片

var img = new Image();

img.src = "img/girl2.jpeg";

img.onload = function () {

//绘制图片

ctx.drawImage(img, 0, 0, myCanvas.width, myCanvas.height);

}

步骤四：实现放大镜效果

添加鼠标事件，获取鼠标的位置，然后放大指定区域的图片，并加上圆形的遮罩，形成放大镜效果。代码如下所示：

myCanvas.onmousedown = function () {

myCanvas.onmousemove = function (event) {

//清除canvas

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

//画图

ctx.drawImage(img, 0, 0, myCanvas.width, myCanvas.height);

//获取鼠标位置

//方式1

var x = event.clientX - myCanvas.offsetLeft;

var y = event.clientY - myCanvas.offsetTop;

console.log(x, y);

//方式2

var x = event.offsetX;

var y = event.offsetY;

//event.offsetX, event.offsetY: 鼠标在事件所关联的元素(event.target)中的位置

//新图部分透明

ctx.globalCompositeOperation = "destination-out";

//画圆

ctx.beginPath();

ctx.arc(x, y, 100, 0, Math.PI \* 2, true);

ctx.fill();

//新图出现在原图的下方

ctx.globalCompositeOperation = "destination-over";

//放大图片

ctx.drawImage(img, 2 \* x - 100, 2 \* y - 100, 200, 200, x - 100, y - 100, 200, 200);

//重置默认值

ctx.globalCompositeOperation = "source-over";

}

}

步骤五：修复bug

当鼠标松开时，把鼠标有移动事件清除，避免出现bug。代码如下所示：

document.onmouseup = function () {

ctx.drawImage(img, 0, 0, myCanvas.width, myCanvas.height);

myCanvas.onmousemove = null;

}

然后运行HTML文件，在输入框里输入名字，并点击确认按钮，即可看到效果。

本案例的完整代码如下所示：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

#my-canvas {

display: block;

margin: 0 auto;

}

</style>

</head>

<body>

<canvas id="my-canvas" width="300" height="450"></canvas>

<script type="text/javascript">

//获取元素

var myCanvas = document.getElementById("my-canvas");

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

//画图片

var img = new Image();

img.src = "img/girl2.jpeg";

img.onload = function() {

//绘制图片

ctx.drawImage(img, 0, 0, myCanvas.width, myCanvas.height);

}

myCanvas.onmousedown = function() {

myCanvas.onmousemove = function(event) {

//清除canvas

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

//画图

ctx.drawImage(img, 0, 0, myCanvas.width, myCanvas.height);

//获取鼠标位置

//方式1

var x = event.clientX - myCanvas.offsetLeft;

var y = event.clientY - myCanvas.offsetTop;

console.log(x, y);

//方式2

var x = event.offsetX;

var y = event.offsetY;

//event.offsetX, event.offsetY: 鼠标在事件所关联的元素(event.target)中的位置

//新图部分透明

ctx.globalCompositeOperation = "destination-out";

//画圆

ctx.beginPath();

ctx.arc(x, y, 100, 0, Math.PI \* 2, true);

ctx.fill();

//新图出现在原图的下方

ctx.globalCompositeOperation = "destination-over";

//放大图片

ctx.drawImage(img, 2 \* x - 100, 2 \* y - 100, 200, 200, x - 100, y - 100, 200, 200);

//重置默认值

ctx.globalCompositeOperation = "source-over";

}

}

1. document.onmouseup = function() {

ctx.drawImage(img, 0, 0, myCanvas.width, myCanvas.height);

myCanvas.onmousemove = null;

}

</script>

</body>

</html>