# 【四、 Canvas开发库封装】

## 《4.1封装常用的绘制函数》

### 4.1.1封装一个矩形

//思考：我们用到的矩形需要哪些绘制的东西呢？

1、矩形的 x、y坐标

2、矩形的宽高

3、矩形的边框的线条样式、线条宽度

4、矩形填充的样式

5、矩形的旋转角度

6、矩形的缩小放大

//下面是把上面所有的功能进行封装的代码：

function XMGRect( option ) {//矩形构造函数

this.\_init(option);

}

XMGRect.prototype = { //矩形的原型对象

\_init: function( option ) { //初始化方法

option = option || {};

this.x = option.x === 0 ? 0 : option.x || 100;

this.y = option.y === 0 ? 0 : option.y || 100;

this.w = option.w || 100;

this.h = option.h || 100;

this.angle = option.angle === 0 ? 0 : option.angle || 0;

this.fillStyle = option.fillStyle || 'silver';

this.strokeStyle = option.strokeStyle || 'red';

this.strokeWidth = option.strokeWidth || 4;

this.scaleX = option.scaleX || 1;

this.scaleY = option.Y || 1;

},

render: function( ctx ) {//把矩形渲染到canvas中

ctx.save();

ctx.translate( this.x, this.y );//位移画布

ctx.rotate( this.angle \* Math.PI / 180 );//旋转角度

ctx.scale( this.scaleX, this.scaleY );//缩放

ctx.fillStyle = this.fillStyle;

ctx.fillRect( 0, 0, this.w, this.h ); //填充矩形

ctx.lineWidth = this.strokeWidth; //线宽

ctx.strokeStyle = this.strokeStyle; //填充样式

ctx.strokeRect( 0,0,this.w,this.h ); //描边样式

ctx.restore();

},

constructor: XMGRect

};

### 4.1.2作业：尝试着封装一个圆形？

function XMGCircle( option ) {

this.\_init( option );

}

XMGCircle.prototype = {

\_init: function( option ) {

option = option || {};

this.x = option.x === 0 ? 0 : option.x || 100;

this.y = option.y === 0 ? 0 : option.y || 100;

this.w = option.w || 100;

this.h = option.h || 100;

this.angle = option.angle === 0 ? 0 : option.angle || 0;

this.fillStyle = option.fillStyle || 'silver';

this.strokeStyle = option.strokeStyle || 'red';

this.strokeWidth = option.strokeWidth || 4;

this.scaleX = option.scaleX || 1;

this.scaleY = option.Y || 1;

this.opactity = option.opactity || 1;

this.counterclockwise =

option.counterclockwise === true ? true : option.counterclockwise || false;

this.startAngle = option.startAngle == 0 ? 0 : option.startAngle || 0;

this.endAngle = option.endAngle == 0 ? 0 : option.endAngle || 0;

this.startAngle = this.startAngle \* Math.PI/180;

this.endAngle = this.endAngle \* Math.PI / 180;

this.r = option.r || 100;

},

render: function( ctx ) {

ctx.save();

ctx.translate( this.x, this.y);

ctx.scale( this.scaleX, this.scaleY );

ctx.rotate( this.agnle \* Math.PI / 180 );

ctx.globalAlpha = this.opacity;

ctx.fillStyle = this.fillStyle;

ctx.strokeStyle = this.strokeStyle;

ctx.moveTo(0, 0);

ctx.arc( 0, 0, this.r, this.startAngle, this.endAngle, this.counterclockwise);

ctx.fill();

ctx.stroke();

ctx.restore();

},

constructor: XMGCircle

};

## 《4.2 第三方库》

百度的echart

http://echarts.baidu.com/echarts2/doc/example.html

国产的egret引擎

http://www.egret-labs.org/

比较火的3d引擎：treejs

FABRIC.JS oCanvas RGraph

# 【Konva】

官网：

<http://konvajs.github.io/>

特点：

\* 小巧、使用方便、适合移动端和pc端

\* 支持丰富的事件处理操作

\* 支持类似JQuery的操作方式

\* 开源，可以随意更改

\* 社区更新比较活跃，github托管源码

\* 性能也不错

# 【五、Konva的使用快速上手】

## 《5.1 Konva的整体理念》

舞台：

整个视图看做是一个舞台 stage

舞台中可以绘制很多个层 layer

layer下面可以有很多的group

group下面可以有 矩形、图片、其他形状等

Stage

|

+------+------+

| |

Layer Layer

| |

+-----+-----+ Shape

| |

Group Group

| |

+ +---+---+

| | |

Shape Group Shape

|

+

|

Shape

## 《5.2 Konva矩形案例》

### 5.2.1 创建一个矩形： Konva.Rect(option);

//Konva使用的基本案例

//第一步：创建舞台

var stage = new Konva.Stage({

container: 'container', //需要存放舞台的Dom容器

width: window.innerWidth, //设置全屏

height: window.innerHeight

});

//第二步：创建层

var layer = new Konva.Layer(); //创建一个层

stage.add(layer); //把层添加到舞台

//第三步： 创建矩形

var rect = new Konva.Rect({ //创建一个矩形

x: 100, //矩形的x坐标，相对其父容器的坐标

y: 100,

width: 100, //矩形的宽度

height: 100, //矩形高度

fill: 'gold', //矩形填充的颜色

stroke: 'navy', //矩形描边的颜色

strokeWidth: 4, //填充宽度

opactity: .2, //矩形的透明度

scale: 1.2, //矩形的缩放 1：原来大小

rotation: 30, //旋转的角度，是deg不是弧度。

cornerRadius: 10, //圆角的大小（像素）

id: 'rect1', //id属性，类似dom的id属性

name: 'rect',

draggable: true //是否可以进行拖拽

});

//创建一个组

var group = new Konva.Group({

x: 40,

y: 40,

});

group.add( rect ); //把矩形添加到组中

//第四步： 把形状放到层中

layer.add( group ); //把组添加到层中

layer.draw(); //绘制层到舞台上

## 《5.3 Konva的动画系统》

### 5.3.1 tween对象(重点)

tween，英文意思：两者之间， 英 [twiːn] 美 [twin]

tween是控制Konva对象进行动画的核心对象。

tween可以控制所有数字类型的属性进行动画处理，比如：x, y, rotation, width, height, radius, strokeWidth, opacity, scaleX等

//案例：

var tween = new Konva.Tween({

node: rect, //要进行动画的Konva对象

x: 300, //要进行动画的属性

opacity: .8,

duration: 1, //持续时间

easing: Konva.Easings.EaseIn, //动画的动画效果

yoyo: true, //是否进行循环播放的设置

onFinish: function() {

//动画执行结束后，执行此方法

}

});

tween.play(); //启动动画

**tween的控制方法**

tween.play(), //播放动画

tween.pause(), //暂停动画

tween.reverse(), //动画逆播放

tween.reset(), //重置动画

tween.finish(), //立即结束动画

seek：英文：寻找 英 [siːk] 美 [sik]

**tween的缓动控制选项**

Konva.Easings.Linear //线性

Konva.Easings.EaseIn //缓动，先慢后快

Konva.Easings.EaseOut //先快后慢

Konva.Easings.EaseInOut //两头慢，中间快

Konva.Easings.BackEaseIn //往回来一点，然后往前冲，汽车启动类似...

Konva.Easings.BackEaseOut

Konva.Easings.BackEaseInOut

Konva.Easings.ElasticEaseIn //橡皮筋 英 [ɪ'læstɪk] 美 [ɪ'læstɪk]

Konva.Easings.ElasticEaseOut

Konva.Easings.ElasticEaseInOut

Konva.Easings.BounceEaseIn //弹跳；弹起，反跳；弹回 英 [baʊns] 美 [baʊns]

Konva.Easings.BounceEaseOut

Konva.Easings.BounceEaseInOut

Konva.Easings.StrongEaseIn //强力

Konva.Easings.StrongEaseOut

Konva.Easings.StrongEaseInOut

### 5.3.2 动画to的使用

to就是对tween的封装，比较简单好用。

//案例：

var rect = new Konva.Rect({

x: 10,

y: 10,

width: 100,

height: 100,

fill: 'red'

});

layer.add(rect);

layer.draw();

//动画系统

rect.to({

x: 100,

y: 100,

opactity: .1,

duration: 3,

onFinish: function() {

}

});

//to: 就是对tween的简单应用。

### 5.3.3 Animate的应用

Animation动画，实际上就是浏览器通知开发者进行绘制，并提供当前的时间

var anim = new Konva.Animation(function(frame) {

//动画系统提供的frame有三个属性可以使用：

var time = frame.time, // 动画执行的总时间

timeDiff = frame.timeDiff, // 距离上一帧的时间

frameRate = frame.frameRate; // 帧率（既1000/间隔时间）

//动画的动作

}, layer);

anim.start();//启动动画

//anim.stop();//结束动画

### 5.3.4 循环播放动画的实现

//总体思路，使用tween 配合onFinish事件中重新播放动画，达到循环播放的效果

var loopTween = new Konva.Tween({

node: star, //设置要表现动画的 Konva对象

rotation: 360, //旋转360度

duration: 2, //动画持续时间

easing: Konva.Easings.Linear,

onFinish: function() {

// this === loopTween //true

this.reset();//重置动画

this.play(); //重新播放动画

}

});

loopTween.play();

### 5.3.5 回放且循环播放动画

yoyo属性可以进行对动画进行播放完后，回放当前动画，并持续循环来回切换播放。

rect.to({

duration: 2,

scale: 1.5,

yoyo: true// 此设置也可以用于 tween

});

### 5.3.6 进度条案例

### 5.3.7 体系环绕案例

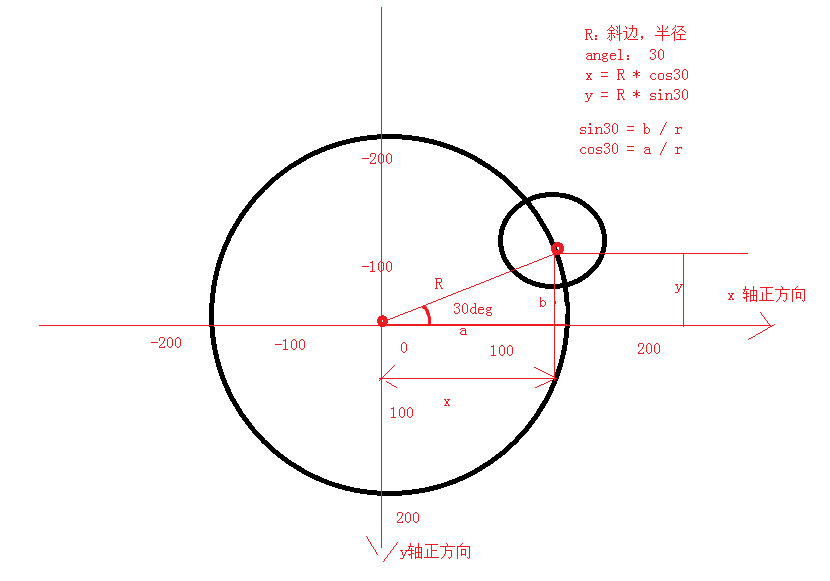
**三角函数的补充**

* Math.sin(弧度); //夹角对面的边 和 斜边的比值
* Math.cos(弧度); //夹角侧边 与斜边的比值

**圆形上面的点的坐标的计算公式**

x =x0 + Math.cos(rad) \* R;//x0和y0是圆心点坐标

y =y0 + Math.sin(rad) \* R;//注意都是弧度



**group的灵活运用**

* konva的group很灵活，每个group都有自己的坐标系
* group可以包含其他的group，可以对group做整个组的动画
* group可以通过getChidren();//可以拿到直接子级元素。

var group = new Konva.Group({

x: 0,

y: 0

});

group.add(rect);

## 《5.4 Konva的事件（重要）》

var rect = new Konva.Rect({

x: 100,

y: 100,

fill: 'red',

width: 200,

height: 200

});

//绑定事件 Konva支持事件：mouseover, mouseout, mouseenter, mouseleave, mousemove, mousedown, mouseup, mousewheel, click, dblclick, dragstart, dragmove, and dragend

rect.on('click', function(){ //jQuery一模一样！！

console.log('^\_^ ^\_^');

});

//绑定多个事件

rect.on('click mousemove',function(e){

});

//解除绑定事件

rect.off('click'); //这不是jQuery吗？

//触发事件

rect.fire('click');

//取消事件冒泡

rect.on('click', function(evt) {

alert('You clicked the circle!');

evt.cancelBubble = true; //取消事件冒泡

});

## 《5.5 Konva的选择器》

### 选择方法：

* ID选择法：stage.find('#id'); //此方法返回的是一个数组
* name选择法：group.findOne('.name');//返回一个Konva对象
* type选择法： group.find('Circle');//查找所有的圆形Konva对象

//组中查找圆形的Konva对象

groupCircle.find('Circle').each(function( circle, index ){

circle.setZIndex( 3 - index );

});

## 《5.6 饼状图案例》

wedge: 楔形，扇形

## 《5.7 柱状图案例》

histogram n. [统计] 直方图；柱状图 英 ['hɪstəgræm] 美 ['hɪstəɡræm]

# 【六、Canvas项目实战】

# 【七、Canvas优化】

<!-- requestAnim shim layer by Paul Irish -->

window.requestAnimFrame = (function(){

return window.requestAnimationFrame ||

window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

window.oRequestAnimationFrame ||

window.msRequestAnimationFrame ||

function(/\* function \*/ callback, /\* DOMElement \*/ element){

window.setTimeout(callback, 1000 / 60);

};

})();

// example code from mr doob : http://mrdoob.com/lab/javascript/requestanimationframe/

var canvas, context, toggle;

init();

animate();

function init() {

canvas = document.createElement( 'canvas' );

canvas.width = 512;

canvas.height = 512;

context = canvas.getContext( '2d' );

document.body.appendChild( canvas );

}

function animate() {

requestAnimFrame( animate );

draw();

}

function draw() {

var time = new Date().getTime() \* 0.002;

var x = Math.sin( time ) \* 192 + 256;

var y = Math.cos( time \* 0.9 ) \* 192 + 256;

toggle = !toggle;

context.fillStyle = toggle ? 'rgb(200,200,20)' : 'rgb(20,20,200)';

context.beginPath();

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

context.closePath();

context.fill();

}