[**pixi.js教程中文版--基础篇**](http://www.cnblogs.com/afrog/p/4056378.html)

**前言**

Pixi.js使用WebGL，是一个超快的HTML5 2D渲染引擎。作为一个Javascript的2D渲染器，Pixi.js的目标是提供一个快速的、轻量级而且是兼任所有设备的2D库。提供无缝 Canvas 回退，支持主流浏览器，包括桌面和移动。 Pixi渲染器可以开发者享受到硬件加速，但并不需要了解WebGL。这里仅是记录项目中使用的经验，即记录利用pixi在制作精灵动画时的一些方法。

=========================================================================

**概述**

需要说明的是，有些示例需要使用http://方式才能有效果，也就是说file://的方式有些示例是无法显示的。

下面我们来看一下，用pixi创建一个精灵要经历哪些步骤：

创建一个PIXI的实例，并展示出来，通常需要以下几步：

1. 创建一个舞台 (stage)
2. 创建一个画布 (render)
3. 把画布插入DOM中
4. 创建一个精灵 (sprite)
5. 把精灵加入画布
6. 把画布加入舞台
7. 刷新舞台

**概念解释：**

 舞台(stage):

// A Stage represents the root of the display tree. Everything connected to the stage is rendered  
所有要渲染的对象都必须连接到舞台中才能被显示出来，舞台处于整个树形展示结构的最底层，可以理解为背景  
  *demo :*

var stage = new PIXI.Stage(0xFFFFFF); //接收一个16进制的值，用于背景的颜色

**画布(renderer):**  
 选用canvas或webGL进行渲染的一个区域  
 demo:

var renderer = PIXI.autoDetectRenderer(400, 300);

它的后面还有两个可选参数，分别 用来指定现有的canvas元素作为容器，canvas背景是否透明，

**材质(texture):**

可以理解成一种承载图片的结构，它本身不能直接用于显示，需要通过精灵(sprite)才能显示，有点类似于dom中的临时碎片（DocumentFragment）  
  
demo:

var texture = PIXI.Texture.fromImage("bunny.png");

**精灵(sprite):**

可以直接用于舞台显示的对象,可以理解为DOM中的element.  
精灵可以直接用图片创建，也可以先创建材质，再用材质创建精灵  
  
*demo :*

var sprite = new PIXI.Sprite.fromImage('assets/image.png');//直接由图片创建  
//...

var texture = PIXI.Texture.fromImage("bunny.png");

var sprite = new PIXI.Sprite(texture);//由材质创建

//  sprite.position.x = 30; //没有单位

//  sprite.position.y = 20; // 这就和给element设置样式是一样的道理。

**事件(event)：**  
PIXI库在精灵和舞台上提供了事件,用于交互.  
 demo:

[复制代码](javascript:void(0);)

stage.click = function(data){

var event = data.originalEvent

}

sprite.click = function(data){

var event = data.originalEvent

var target = event.target

//阻止默认行为,sprite.buttonMode = false;

event.preventDefault();

}

[复制代码](javascript:void(0);)

 此外，还有.mousedown,.mouseover,.mouseout,.mousemove,.mouseup ,.mouseupoutside , .touchstart,.touchend,.tap

**注意事项：**

1. 对于有事件的精灵，通常需要设置sprite.interactive = true ， 对于作按钮用的精灵，需要设置sprite.buttonMode = true;
2. 与DOM的事件不同的是，精灵中的事件，是进行了二次封装的，
3. 通常需要用var event = data.originalEvent来与普通dom中的event对象保持一至

 不过又不完全相同，如data.originalEvent.stopPropagation()就不起作用.

 关于stopPropagation的问题，我问过pixijs的开发者，他答复说stopPropagation是dom中的概念，在canvas/webGL中是没有的，所以不能说是pixi的问题。

**下面使用PIXI库显示一张图片,简单的演示下pixi的用法：**

[复制代码](javascript:void(0);)

var stage = new PIXI.Stage();

var renderer = PIXI.autoDetectRenderer(400, 300);

document.body.appendChild(renderer.view);

var sprite = new PIXI.Sprite.fromImage('bunny.png');

stage.addChild(sprite);

[复制代码](javascript:void(0);)

 直接贴出上面的代码，在网页上只能看到漆黑的背景， 这是因为精灵的创建是异步的，必须等加载完图片之后才能把精灵加入舞台中  
   
我在这里给出两种解决方式：

 1：不断的去刷新舞台：

requestAnimFrame(animate);

function animate(){

renderer.render(stage);

requestAnimFrame(animate);

}

2. 在图片加载完成的回调中执行。这仅限于不需要刷新的精灵、

sprite.texture.baseTexture.on('loaded',function(){

renderer.render(stage);

})

在项目中实践的时候发现用基于baseTexture的loaded事件有两个问题  
1：如果图片加载失败怎么办？  
2：同时加载多张图片的时候，有些没有触发图片的loaded回调  
  
经过实践，找到一个更加稳妥的办法：

[复制代码](javascript:void(0);)

var img = new Image();

img.src = 'bunny.png';

img.onload = function(){

var baseTexture = new PIXI.BaseTexture(this);

　　var texture = new PIXI.Texture(baseTexture);

var sprite = new PIXI.Sprite(texture);

stage.addChild(sprite);

renderer.render(stage);

}

[复制代码](javascript:void(0);)

此方案对于批量加载图片也是适用的。最后再重复一遍，请用localhost://xxxxxx的方式进浏览

细心的比较也许你会发现，我和官方的示例不相同，官方使用了requestAnimFrame( animate )，而然我这里没有用，为什么？因为我只是要显示一张图片而已，没有必要不断的去刷新舞台。requestAnimFrame( animate );这个是在精灵有变化的时候才需要用到。当然，如果不考虑性能，对一张静态的图不断的刷新也没有什么问题。

完整的示例：

[复制代码](javascript:void(0);)

var stage = new PIXI.Stage(0xFFFFFF);

var renderer = PIXI.autoDetectRenderer(400, 300);

document.body.appendChild(renderer.view);

var sprite = new PIXI.Sprite.fromImage('bunny.png');

stage.addChild(sprite);

sprite.texture.baseTexture.on('loaded',function(){

renderer.render(stage);

})

sprite.interactive = true

stage.click = stage.tap = function(data){

console.log(data.originalEvent.target);

//alert('you hit the stage')

}

sprite.mousedown = sprite.touchstart =function(data){

// stop the default event...

data.originalEvent.preventDefault();

this.data = data;

this.alpha = 0.9;

this.dragging = true;

console.log('mousedown');

}

sprite.mouseup = sprite.mouseupoutside = sprite.touchend = function(data){

this.alpha = 1

this.dragging = false;

// set the interaction data to null

this.data = null;

console.log('mouseup');

};

sprite.mousemove = sprite.touchmove = function(data){

if(this.dragging){

var newPosition = this.data.getLocalPosition(this.parent);

this.position.x = newPosition.x;

this.position.y = newPosition.y;

console.log('mousemove')

renderer.render(stage);

}

}

//requestAnimFrame(animate);

function animate(){

renderer.render(stage);

requestAnimFrame(animate);

}

[复制代码](javascript:void(0);)

精灵放大2倍

[复制代码](javascript:void(0);)

var sprite = new PIXI.Sprite.fromImage('bunny.png');

var sx = sprite.scale.x;

var px = sprite.position.x;

sprite.scale.x += 1;

sprite.position.x -= px;

//Y方向同理

...

//刷新一下

requestAnimFrame(animate);

[复制代码](javascript:void(0);)

精灵的层级问题

在canvas中，由于不能用css的层级样式表来控制层级关系，所以只能另想办法进行模拟。pixi.js中没有提供现成的方法，不过我们研究发现，只要控制精灵加入画布的顺序就可以实现类似的层级效果。它的规则是这样的，先加入画布的，出现在最下面，后加入的，出现在最上面。于是自然的就想到了先用一个数组收集这些精灵，排好序之后再循环加入画布。可是问题就来了，有时候，我们事先往往不能确定层级，或者有时候是用户触发某一个动作之后才改变层级的，这怎么实现呢？这个我想到了给精灵人为的加一个zindex属性。然后遍历这个属性，进行排序，最后再刷新一下舞台。

[复制代码](javascript:void(0);)

　　　　/\*\*

\* 对象按指定的属性排序

\* @param {array} arr 对象集合

\* @return null

\*/

function pixiSort(arr){

var ln = arr.length;

var tmp;

for(var i = 0;i<ln-1;i++){

for(var j=i+1;j<ln;j++){

tmp = arr[i];

if(arr[i].sortIndex > arr[j].sortIndex){

arr[i] = arr[j];

arr[j] = tmp;

}

}

}

}

[复制代码](javascript:void(0);)

受html dom操作的影响，有时候我们需要精灵按照指定的层级进行排列显示，比如给精灵指定index属性。要做到对zindex的控制，可以用到如下方法

[复制代码](javascript:void(0);)

function updateZindex(ob,stage){

Object.defineProperty(ob,'zIndex',{

set : function(value){

if(this.sortIndex !== value){

this.sortIndex = value;

pixiSort(stage.children);

}

},

get : function(){

return this.sortIndex;

}

});

}

[复制代码](javascript:void(0);)

 说白了，就是控制精灵的先后顺序来达到显示层级的效果。

最后是如何销毁：

//移除全部精灵

stage.removeChildren()

//移除整个舞台的引用

stage.removeStageReference()

以上创建的是建静止的精灵，及在精灵上邦定事件。这个在官方没有找到示例，特补充一下。（或许pixi就是为动画而生的，所以没有考虑这种简单的需求了）

如果需要动画效果的精灵，官方提供了多种方法。这里就不再细述了。我目前正在研究这一块，有兴趣的可以和我（@frog）一起交流！

**在线资源**

Pixi.js的GitHub主页上，列举了很多在线Demo。

GitHub：<https://github.com/GoodBoyDigital/pixi.js/>

官网： http://www.pixijs.com/

API :http://www.goodboydigital.com/pixijs/docs/

官方教程：http://www.goodboydigital.com/pixi-js-tutorial-getting-started/

https://github.com/GoodBoyDigital/pixi.js 打包下载后，解压，在example里边有23个应用示例。