# jsPlumb介绍

**ShenBY 沈本义(qq:1617309239)2011-11-07**

**个人博客:www.shenbenyi.cn**

以下jsPlumb介绍基于jsPlubm1.3.3版本，并且基础库是用jQuery1.3.x或以上，与其他版本基础库或jsPlumb的比较或后期版本升级，本文档不做介绍。

本人英文水平有限，有疑惑请自行到官网对照翻译。

## 摘要

Jsplumb是Jquery的一个插件，它能够让你用动态的或静态的链接来连接html界面上行的元素，并且从1.1.0版本开始，提供用鼠标拖动来链接。目前该插件支持三个javascript库，有Jquery、MooToos、Yui3，jsplumb代码是开源的，并且是麻省理工学院许可，由google进行代码托管。

官方示例：<http://morrisonpitt.com/jsPlumb/html/demo.html>

代码地址：<http://code.google.com/p/jsplumb/>

Jsplumb介绍地址：<http://jsplumb.tumblr.com/>

JsPlumb允许您使用SVG、Canvas 或者 VML链接屏幕上的元素，这些取决于您使用的浏览器的能力。

**浏览器的兼容性**

jsPlumb 1.3.3 已经在以下浏览器测试：

IE 6 on Windows XP

IE 7 on Windows XP

IE 8 on Windows XP

Firefox 3.5.8 on Windows XP

IE 9 on Windows 7

Chrome 12 on Windows 7

Firefox 3.5.8 on Windows 7

Firefox 3.6.3 on Ubuntu 10.04

Chrome on Ubuntu 10.04

Safari 4 on Mac Tiger

Safari 4 on Windows Vista

Safari 5.0.5 on Windows 7

Opera 10.54 on Windows XP

## 安装

### 需要导入

使用jsPlumb需要到如的类库文件，是根据您使用的javascript类库的不同而定，目前提供以下版本。

**jQuery**

jquery的1.3.x或更高版本

jquery ui的1.7.x或1.8.x(如果您需要支持拖放)



**MooTools**

MooTools核心库版本1.2.4或更高版本(jsPlumb已在1.2.4和1.3.3版本上测试)

MooTools的Drag.Move的1.2.4.4或更高版本



**YUI3**

YUI3.3.X ,目前jsPlumb只在YUI3.3.0上测试，YUI3其他版本可能也可以正常工作



jsPlumb可以使用SVG，HTML5的画布元素或VML去呈现需要显示的对象，现代大部分浏览器都支持Canvas 和 SVG,但是IE9以下都不支持这些，默认情况下，jsPlumb都使用canvas去渲染，但如果在使用IE9以下的情况下，jsPlumb将使用vml去渲染，您可以通过下列方式调整渲染模式：



## 二．简介

### 1. 基本概念

jsPlumb是将所有链接的东西放在一起，所以在jsPlumb中，核心是链接对象，jsPlumb本身可以分成四个组成部分

**锚(Anchor)**：一个位置，放置端点的地方，相对于一个元素的来源，您不需要自己硬编码来创建它，jsPlumb提供给您各种功能，您只需要按照您的需要创建它就可以了。它没有可视化的显示，只是一个逻辑位置，可以使用锚的id来引用它，jsPlumb支持这样做，并且您可以使用坐标来表示[x,y,x方向,y方向]

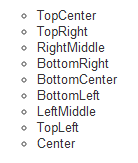
**端点(Endpoint** **)**：链接的一端的可视化表示，您可以创建并可以链接他们；您可以让他们支持拖拽，或者您可以直接使用jsPlumb.connect()在创建链接时直接创建它们。

**连接器(Connector)**：链接两个元素的线，页面的可视化表示，jsPlumb有三种默认类型:Bezier曲线，直线，和流程图链接器，您不用去处理连接器，当您需要使用它们时，您只需要定义它们即可。

**覆盖物(Overlay)**：一个UI组件，是用来是用来装饰连接器，例如标签、箭头等。

### 锚(Anchors)

锚的概念是指：定义一个链接线能够链接的点，jsPlumb有9个默认的锚点位置，您可以使用它们去链接元素，具体有：



这些位置在jsplumb底层代码中都是以阵列语法表示的，[X,Y,DX,DY],其中X,Y是在区间[0,1]指定锚的位置，DX和DY是在区间[-1,1]指定曲线的事件锚的方向坐标，例如[0，0.5，-1，0]定义了一个“LeftCenter”连接器的曲线，从锚点单向向左的曲线.同样，[0.5，0，0，-1]，定义一个“CenterTop”锚连接器所产生的向上的曲线。

### 动态锚(Dynamic Anchors)

这些都是可以在若干地点之一定位的锚点，当你每次移动一个元素时，会自动选择一个最合适的位置，没有特殊的语法来创建一个DynamicAnchor，你只需要提供一个独立的锚位置，例如数组：



默认的动态锚：

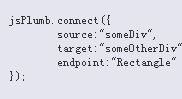
jsPlumb提供一个动态的锚，定名为“AutoDefault”默认位置有：TopCenter，RightMiddle，BottomCenter和LeftMiddle



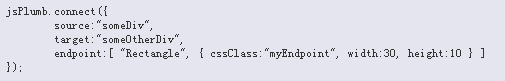
动态锚和可拖动的连接是可以进行互操作的，当你开始拖动连接或释放它时，jsPlumb会锁定一个动态的锚的位置，您可以在界面上看到链接锚点的切换变化。

### 连接器、端点，覆盖物(Connector, Endpoint & Overlay Definitions)

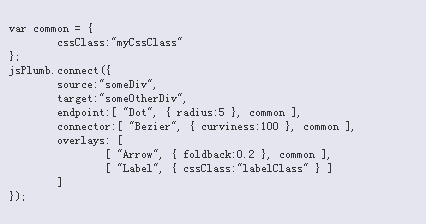
在我们讨论连接器、终点和覆盖物之前，需要提醒的是：你需要定义一个连接器，端点或覆盖，您必须使用一个“定义”，而不是直接构建一个，这个定义可以是一个字符串，它指定你需要创建的对象。



或者您需要构件的对象的属性名组成的数组，您可以通过其构造函数来创建，例如



也有三个参数的方法，可以让你指定两套参数，jsPlumb会自动为你合并，例如



### 连接器(Connectors)

链接的线，实际上是界面上的各个元素的链接线，jsPlumb有三条连接器实现，一条直线、Bezier曲线和”流程图”连线，默认的连接线是贝塞尔曲线，您可以有选择的设置一个链接器，可以通过设置“connector”来定义一个连接线，或者在添加端点时设置连接线，如果您没有为connector设置一个值，那么他会用他的默认值”Default”

三种链接的定义语法类似

**贝塞尔曲线**：

贝塞尔曲线提供了两个端点之间的立方体路径，它支持一个构造函数参数，

**curviness** -可选，默认为150，这定义的锚点位于贝塞尔曲线的控制点，以像素为单位的距离，但这并不意味着你的连接器会穿过从你的曲线到这个距离的点，这仅是一个标识而已。

**直线**：直连绘制直线的两个端点之间。 没有构造函数的参数支持，使用参数endpointStyle定义连接样式或添加端点时定义连接线样式

**流程图**：这种类型的连接器，绘制一系列垂直或水平段组成的连接 - 经典的流程图​​，支持一个单一的构造函数参数

**存根** -这是最小长度，以像素为单位，从端点发出的初始存根。 此参数是可选的，默认为30像素

### 端点类型(Endpoint Types)

一个端点的UI组件，标志着一个锚的位置，是连接器连接的点，jsPlumb有三个端点实现，点、矩形和图形，你可以指定端点的属性，在jsPlumb.connect时指定属性，或者jsPlumb.Addendpiont时指定，或者设置jsPlumbde target时设定属性，语法可参照connector。

三个可用的端点类型，他们的构造函数参数如下：

**点端点(Dot Endpoint)**：

此端点在屏幕上画一个点。 它支持一个构造函数参数：

**半径** -可选，默认为10个像素。 定义点的半径

**矩形端点(Rectangle Endpoint)**：

绘制一个矩形。 支持的构造函数的参数是：

**宽度** -可选;默认为20像素。 定义矩形的宽度。

**高度** -可选，默认为20个像素。 定义矩形的高度。

**图片端点(Image Endpoint)：**

从一个给定的的URL绘制图像。 此端点支持一个构造函数的参数：

**SRC** -必需的。 指定要使用的图像的URL。

### 覆盖物类型(Overlay Types)

覆盖界面上的链接元素，如标签或箭头等，jsPlumb有四个默认值：

**Arrow**：可配置的箭头，放在两个点的连接线上，你可以控制箭头的长度和宽度，

转折点：是一个折回点，方向点(允许值是1和-1，意味着是点的链接方向)

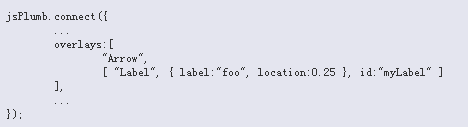
**Label**：在点的连接器上的可配置的标签

**PlainArrow：**一个三角形箭头，没有折回点

**Diamond:**顾名思义，钻石

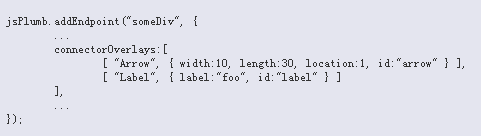
最后的两个实际上市Arrow的配置的实例，当您调用jsPlumb.connect或jsPlumb.addEndpoint时，您可以指定一个或多个覆盖物。

1. 这里我们举个例子，默认的箭头，上面加上文字”foo”，如图：



这个链接将有一个箭头标签在连线的一般，并且文字标签在连线的四分之一处，可以注意下id属性，你可以用它来删除标签或做其他操作。

1. 下面的实例，添加两个覆盖物，请注意，我们在这个例子中使用的参数“connectorOverlays，”而不是上面例子中的“overlays”，这是因为总有一天overlays会用来作为终结点的覆盖物样式：



这个链接将有一个10\*30的的箭头和含有文字foo的标签，位置在连接线的开头，另外，需要注意每一个覆盖物的id。

1. PlainArrow

这仅仅是一个特殊的Arrow的实例，内部jsPlumb给”foldback”硬编码为1，意思是一个平面的箭头尾部边缘，Arror的所有构造函数适用于PlainArrow.

1. Diamond

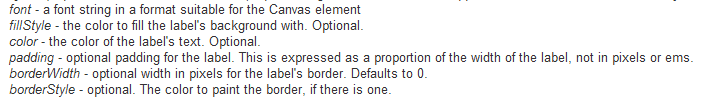
这也是一个特殊的Arrow实例，其中jsPlumb内部给foldback硬编码值为2，意味着箭头转变为Diamond,Arrow的所有构造函数适用于Diamond.

1. Lable

提供了一个文本的标签来装饰连接器，可用的构造函数参数有：

Lable：要显示的文字，你也可用提供一个函数放这儿，用来替代文本，他是做为连接的参数返回，应该返回一个string类型的字符串；

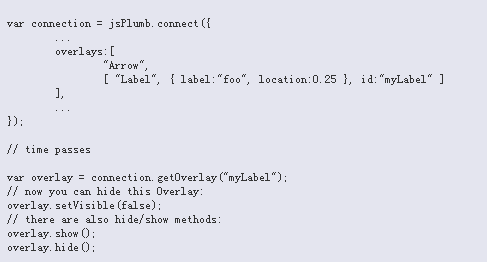
cssClass：一个可选的用于lable的样式类，这是目前首选的”lableStyle”参数，样式有：



Location：做为“Arrow”标签应该显示的位置,比例从0到1，包括1。

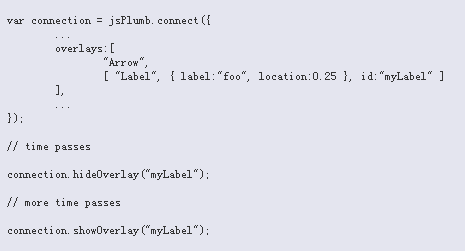
1. 显示/隐藏覆盖

你可用使用setVisible来控制Arrow的显示或隐藏，或者showOverlay（ID）方法，或者hideOverlay（ID）方法，请注意这里的参数id，是我们在设定各个元素时指定的，它可用用来找到那个元素。简单的实例：

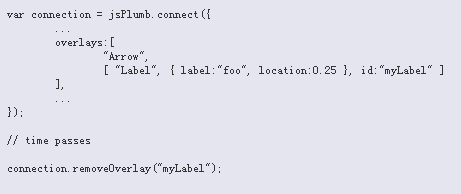


并且，连接器connection也有很方便的方法用来替代上面的方法

如图：

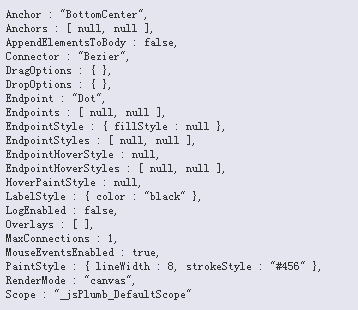


连接器connection也提供删除覆盖物的方法，如图：

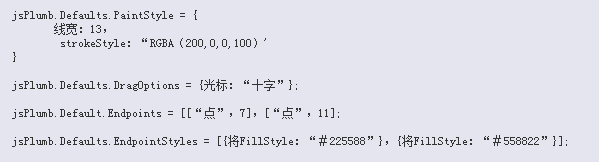


### 默认值(Defaults)

最简单的方式来设置你的pluming的显示外观等等，是重写这些元素的默认值，如果你不这么做，你得每次都手动的去提供这些值；每一个connect和addEndpoint方法都有一个关联的默认值，这些默认值存储在jsPlumb.Defaults里面，他是一个javascript对象，这些参数的初始值是：



您可用用类似下面的方法来覆盖默认值



## 三．链接(Connections)

### 1.基本链接(Programmatic Connections)

1. 最简单的链接，你可用这样写



在这个例子中，我们已经创建了“element1的”element2的“连接。 请记住，在jsPlumb中连接是靠两个端点，连接器，和零个或多个覆盖物重叠组成，但是，这个实例的“连接”没有提供这些东西，所以jsPlumb在需要这些的地方都用的是默认值，这些默认值是：

--链接的两个端点，jsPlumb默认使用的是Dot点端点，半径10，填充颜色#456

--链接的两个端点的锚，也是用的是默认值

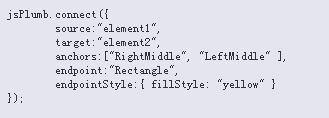
--设定两个端点是否可被用来做为新的链接点，默认值为false

--链接器的类型和外观，默认值是贝塞尔曲线，宽度8，颜色#456

所以这个链接是：

一个8px的贝塞尔曲线，从一个端点(element1)的bottomCenter移到另一个端点的bottomCenter，而且每一个端点的都是以10px为半径，#456为颜色的Dot点，

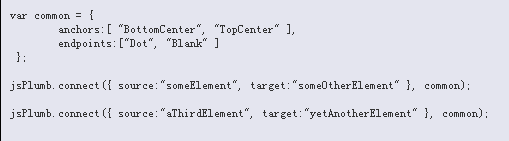
1. 让我们稍微加强一点这个链接，并且告诉jsPlumb我们想要什么，如图：



具体样式…就不介绍了，自己看图。

1. 重用jsPlumb的链接设置

一种非常常见的情况，比如你想要同一个设置，来创造多个链接，jsPlumb connect有两个参数



### 2.使用拖放的链接(Programmatic Connections)

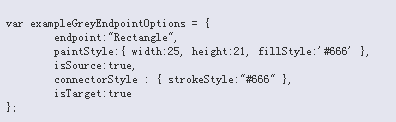
如果需要链接支持拖放，你首先需要进行一下设置，每一个需要支持拖放的链接，至少需要一个终结点，这里给一个实例，展示怎样去创建Endpoint

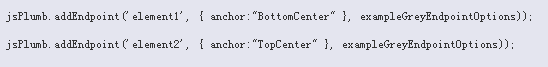


此端点将作为连接的资源，任何来自他的链接都将使用jsPlumb默认的样式作为自己的样式

Tip:一些常用的数据使用三个参数的addEndPoint方法

经常发生的一件事：你有一个端点，他的样式和其他的端点只有一点点区别，大多数都是一样的，这样的情况下，你可以定义个相同的样式，有不同的样式可以单独立出来，如下图，注意下anchor设置，两个元素是不一致的：





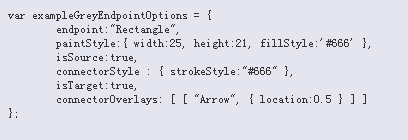
现在你有一个源端点，你需要在其他元素上创建一个终结点，或者通知jsPlumb你需要使整个元素成为一个拖拽的目标，让我们看看怎样去创建这样一个端点：



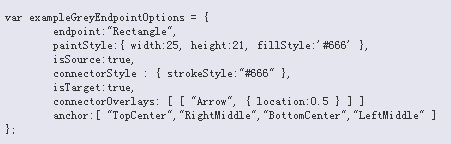
还有另外一个方法，jsPlumb.makeTarget()，这个方法需要两个参数，第一个参数是指需要设定的元素，或者元素列表，第二个参数是指当一个链接建立时你需要的端点，在这个例子中，我们用我们之前就用过的端点，-the gray Rentangle—但是我们需要告诉jsPlumb,元素” aTargetDiv”将会被作为dorp的端点：



在jsPlumb中方法**jsPlumb.addEndpoint** 和方法**jsPlumb.makeTarget，对象**endpointOptions里面的参数都是一样可以用的，你可以在端点终结点设置很多东西，这个你可以参见api列表说明。下面的示例显示了指定的箭头在链接拖动到终结点的连接线的一半的地方



下面这个示例展示的是，这个终结点，连接到它的连接线是依赖这个终结点的位置，因为它定义了好几个锚点，连接线会自动找到该终结点的动态锚：



### 3.”拖”选项(Drag Options）

这些拖动选项将通过jsPlumb api库来提供支持，jsPlumb将会提供你需要的一切，如果你需要一些拖动过程中的生命周期事件，你需要提供给jsPlumb一个通知方法，如果触发某个事件，它会调用该方法，比若说你需要在拖动开始时：



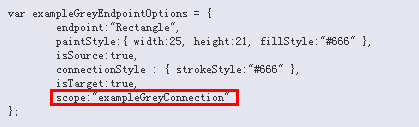
### 4.”放”选项(Drop Options)

Drop选项和Drag方式是一样的，用法可参见jquery用法

<http://docs.jquery.com/UI/Droppable>

### 5.”拖放”范围(域)( Drop Options)

jsPlumb借用了jquery里面drag/drop的范围概念，范围的概念是指那些可拖放的元素只能够被拖放到那些有范围的元素里面，在jsPlumb中你可以在创建EndPoint时创建一个范围”Scope”，下面给出一个例子：

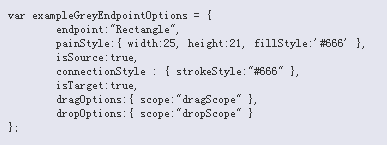


如果你不提供一个”Scope”，jsPlumb将使用一个默认的范围，在这个方法中，这个值是可用的：

如果你因为某些原因想改变它的值，你可以这样做：



如果你想，你也可以这样，提供一个drap和drop的范围值：



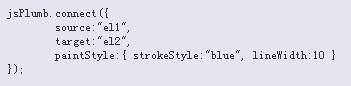
## 四．外观

### 1.基本样式

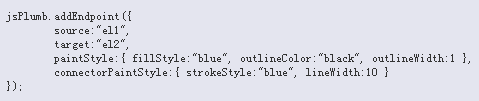
定义端点、覆盖物(覆盖物现在更倾向于用cssClass来替代)、连接线的样式，这些样式是由paintStyle作为jsPlumb connect、jsPlumb.makeTarget、jsPlumb.addEndPoint方法的参数来定义的，根据你调用的不同方法，参数名称会不一样，具体不同如下面所列：

**连接线样式**

**jsPlumb.connect:**



**jsPlumb.addEndpoint:**

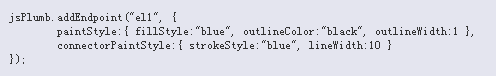


注意上面的样式参数，下面我们对它进行讨论

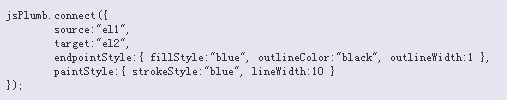
**终结点样式**：

像上面的定义我们可以有下列方法：

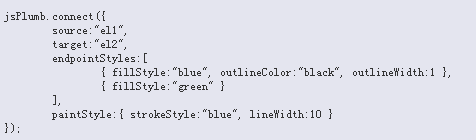
1. 添加终结点时定义：



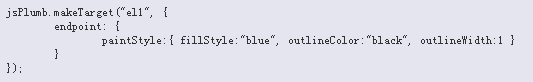
1. 作为jsPlumb.connect连接时的参数



1. 或作为jsPlumb.connect连接对象样式数组中的一个



1. 或者在jsPlumb.makeTarget方法中



在上面的示例中，我们讲”e1”拖放到作为目标的”e2”中，并且定义了终结点的样式。

**覆盖物样式**

最完美的设置覆盖物样式的方式是使用创建overlay时的构造函数来定义

**样式参数列表**

这是设置paintStyle的完整参数列表

**但是请注意，fillStyle参数在connector中会被忽略，strokeStyle在EndOptions中会被忽略，此外，如果你使用jsPlumb.connect创建链接，并且为指定任何EndOption样式，EndOptions中的fillStyle样式会被指定为连接线的strokeStyle。**

fillStyle, strokeStyle and outlineColor可以使用任何有效的CSS3语法

**fillStyle**:定义EndPoint的颜色，例如rgba(100,100,100,50), "blue", "#456", "#993355", rgb(34, 56, 78)

**strokeStyle:**连接器的颜色

**lineWidth:连接线的宽度，是个Int值**

**outlineWidth：连接器或端点的轮廓宽度，int值**

**outlineColor：连接器或端点的颜色**

**dashstyle、The VML spec**、**stroke-dasharray、stroke-dashoffset、stroke-dashoffset、stroke-linejoin**

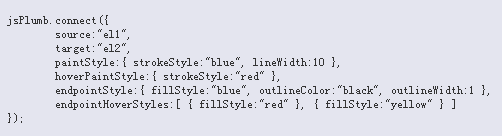
**只适用于VML或SVG或VML和SVG，这里不做介绍**

### 2. 悬浮样式

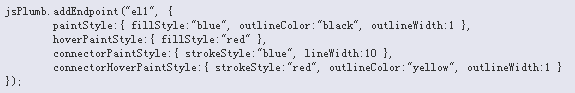
当鼠标悬浮在这些元素上时，连接线和终结点都支持鼠标悬浮样式。他们的样式需要被精确的指定，指定的方式和我们在上面讨论的方式一致，悬浮样式也一样继承了基本样式，这是因为当鼠标移上去的时候，您通常会只希望更改颜色，或者轮廓颜色，所以你只需要指定需要改变的值即可，这样避免了你需要定义重复的样式，鼠标悬浮样式的参数名只是需要在正常样式前加入”hover”即可，下面给个例子：



在这个示例中，我们指定两个链接线和终结点的鼠标悬浮样式，这地方的两个连接线和终结点的悬浮样式是一样的，我们也可以使用复数形式使两个端点或两条连接线的悬浮样式不一样，如下图：

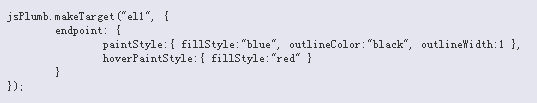


上面示例中两个终结点的鼠标悬浮样式是不一样的。通过下面的两种方式也可以实现， **jsPlumb.addEndpoint和jsPlumb.makeTarget** ：



（这里我们指定了端点的样式和所有从这个端点链接的线的样式）

和



（makeTarget也同样支持paintStyle样式）

但是我们得注意，makeTarget不支持连接线的参数，它只支持终结点。所有的链接中的连接线的参数是当你在用链接方法时在源端点设置的。

### 3. Gradients渐变

Canvas 和SVG 都支持渐变，VML不支持，jsPlumb用它自己的语法来定义渐变，来适用于SVG 和Canvas 之间不同的语法。

有两种类型的渐变可用：

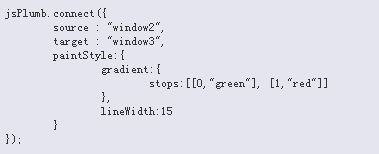
(Linear) 线条渐变：彩色线条都在同一个方向

(radial) 半径的，光线似的渐变：有彩色圆圈组成，有这个圈到那个圈的样式。

由于他们的基本样式不同，jsPlumb仅为连接线提供了Linear(第一种)渐变，但为端点提供了两种方式。

**线条渐变**：

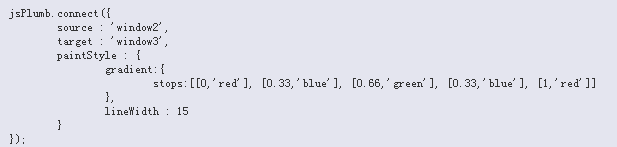
要在链接中使用渐变，你需要提供一个渐变的对象，例如：



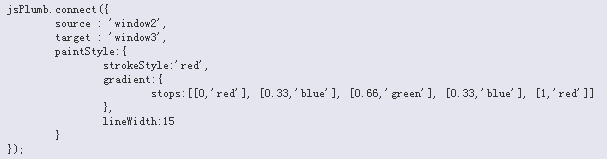
这里我们有个从win2到win3的连接线，宽度15px;颜色是从绿色到红色的渐变。

注意这地方的渐变对象和stops列表，渐变对象里面的颜色梯度可以是任意数值，每中颜色梯度包含两个值[position,color]，position值得范围必须在0到1之间，数值类型是decimal，表示该种颜色开始的位置，颜色的有效值和strokeStyle是一样的。

Stops列表可以包含任意个数，下面的例子表示颜色从红-蓝-绿-蓝-红



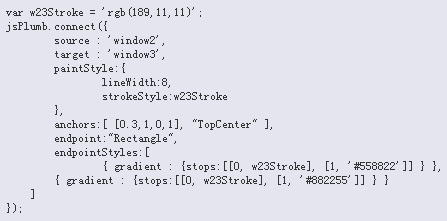
如果你使用VML，jsPlumb将会忽略你的渐变指令，所以最好的做法是，你需要单独指定strokeStyle ，因为默认情况下，如果不支持gradient时，jsPlumb就会返回使用使用gradient的基本样式strokeStyle ，如果你没提供strokeStyle ，则默认都是黑色的，所以最好还是提供strokeStyle 样式，例如：

****

**端点渐变**

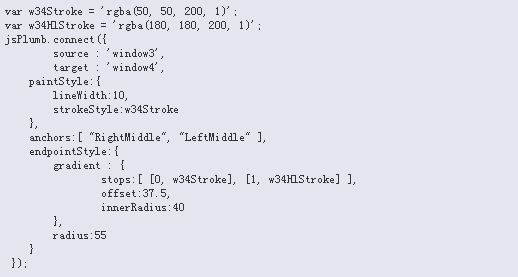
端点的渐变样式语法和连接线的语法是一样的，你指定端点的渐变样式要么放在endPoint的对象设置里面，要么放在连接线定义时，不同端点的定义数组里面，并且，该信息适用于您正在创建的需要拖拽的端点定义。

看下面示例：



只有Dot和Rectangle 类型的端点支持，图片端点时不支持的，当然在VML中也不支持。

Dot示例：

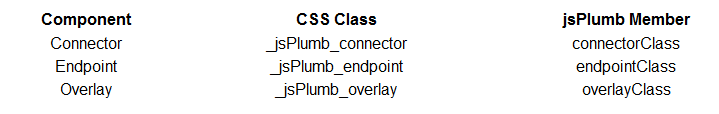
显示一个端点，颜色渐变从最外端开始，端点的渐变需要的数据显然会比线条渐变多，在一个线条的渐变中，我们只需要从一个点移动到下个点即可，但是在端点的渐变中，我们需要从一个圈移动到另一个圈，默认情况下，jsPlumb会使用一个相同半径的圈。

并且offset和innerRadius也可以使用百分数来替代。

## 五．杂项

### 1. CSS样式

jsPlumb在它创建的每个组件时都会附加一些类，这些类名都是公开的，并且如果你需要的话你可以将他们重写，具体如下：



你可以运用这些样式类来使你的界面达到最佳效果。

### 2. Animation动画

jsPlumb提供动画的方法，它是将你使用的基础库中的动画引擎进行封装，并且提供一个回调函数，以供jsPlumb在每一步进行跟踪，你可以自己自己定义这个回调函数，非常方便：



回调函数参数解释：

el:元素id,或者元素的对象

properties:动画的属性，例如持续事件等

options:动画选项，例如回调函数等。

### 3. 找到链接信息

jsPlumb提供了一个完全公用的方法，getConections—来找到关于当前管理的链接，在用这个方法之前，你需要了解jsPlumb中scope概念，

1. 检索单个范围的链接
   1. 获取jsPlumb默认的“scope”中的链接



* 1. 获取“scope”为myScope中的链接



1. 更进一步的过滤

getConnection方法提供含js对象的方法来筛选，可有如下三个对象：

scope:您要检索的链接所属范围(域)

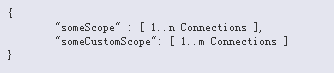
source:返回的链接限制为那些有此源id的链接

target:返回的链接限制为那些有此目标端点id的链接

提供的这三个参数，可以作为对象，也可作为字符串来传递



（返回的是在这两个scope中的连接）

返回值是：

下面列出不同的获取示例：

1. 获取所有链接



1. 获取默认scope的链接



1. 获取指定scope的链接



1. 获取指定的多个scope的链接，返回



1. 返回指定source的链接



1. 返回指定target的链接



1. 返回指定source和指定target的链接



1. 三个参数并用



## 六．事件

jsPlumb支持绑定到不同的事件，有连接线、端点、覆盖物和jsPlumb本身的事件，如果需要绑定时间，你需要使用一个方法：jsPlumb.bing(object,event,callback)，但是这除了overlay的事件，因为overlay的事件可以直接在overlay定义时直接绑定overlay的事件监听，

### 1. jsPlumb事件

jsPlumbConnection—当设定一个连接时会通知，这个新连接对象将会作为回调函数参数传递

jsPlumbConnectionDetached—当一个链接被取消时通知，这个被取消的链接对象会被作为回调函数的参数传递

Click—获取链接的点击事件，回调函数将传递点击的链接和鼠标事件

Dbclick—双击，其他同上

Endpointclick—当端点点击事件时通知，端点对象和鼠标事件是回调函数的参数

endPointDbClick—双击，其他同上

### 2. Conection事件

Click—当一个链接被点击时触发，回调函数传递的参数是鼠标事件和connection对象

Dbclick—双击，其他同上

Mouseenter—当鼠标进入链接的path触发，回调函数传递的也是鼠标事件和connection对象

mouseExit—当鼠标从connection的path离开时触发，其他同上

### 3. Endpoint事件

Click-- 端点被点击时触发，回调函数传递的参数有鼠标事件和端点对象

Dbclick—双击事件时触发，其他同上

mouseEnter—鼠标进入端点时触发，回调函数传递的也是鼠标事件和端点对象

mouseExit—当鼠标从端点中离开时触发，回调函数传递的参数也是鼠标事件和端点对象

### 4. Overlay事件

Overlay的事件监听注册有点不同，你将他们作为overlay的构造函数参数，这是因为你从来不会对overlay有真正的动作，下面的示例看看怎么样注册一个overlay的事件：

jsPlumb.connect({

source:"el1",

target:"el2",

overlays:[

[ "Label", {

events:{

click:function(labelOverlay, originalEvent) { alert("you clicked on the label overlay for this connection :" + labelOverlay.connection) }

}

}]

]

});

## 七．示例

### 1. Connection示例

jsPlumb.connect示例、拖拽的connection示例参见文档<http://jsplumb.org/doc/usage.html>

### 2. 使用工具函数

1. 分离win5和其他元素的所有链接



2. 隐藏win5的所有链接



3. 隐藏win5链接的所有链接和终结点



4. 显示所有win5的链接



5. 显示win5的所有链接和终点，请注意这个示例中的两个参数，如果它决定这个链接中的其他端点也是不可见的，jsplumb也不会使这个链接显现。



1. 切换win5的链接线的显示或隐藏



1. 重画win5的所有链接



1. 重画win5,win6,win11的所有链接



1. 重画所有链接



1. 取消所有链接



1. 删除从win1出发到给定终结点中间的所有链接和终结点



1. 删除和win1链接的所有端点和之间的链接



1. 删除jsplumb实例管理的端点，删除他们之间的链接，这是和jsplumb.reset()方法相同的地方，但它不会清除时间监听列表。



1. 删除指定的端点和它的连接线



1. 删除所有的端点、端点、并且清除事件监听列表，返回jsplumb实例的初始化形态



1. 设置端点不可拖动，不管怎样发出指令



1. 设置两个端点不可以拖动：



1. 设置默认情况下是否可拖动元素，默认情况下是可以拖动的



1. 设置win1可拖动





1. 设置win1和win2都可拖动



1. 设置所有包含class=”windows”的元素可拖动

（jQuery下，其他js库下面不列出）

## 八．开发

目前jQuery-jsplumb-1.3.3-all.js包含了下面的js库集合：

jsPlumb-1.3.3.js

jsPlumb-defaults-1.3.3.js

jsPlumb-renderers-canvas-1.3.3.js

jsPlumb-renderers-svg-1.3.3.js

jsPlumb-renderers-vml-1.3.3.js

jquery.jsPlumb-1.3.3.js

jsBezier-0.2-min.js