[可视化D3] 理解数据绑定

当你想要使用D3绘制散点图时，你需要在图中创建多个`circle`来表达你的数据信息。这时你会发现D3根本没有给你一个创建多个DOM元素的原生方法。

虽然它为我们提供了`.append`这个方法来创建单个元素。

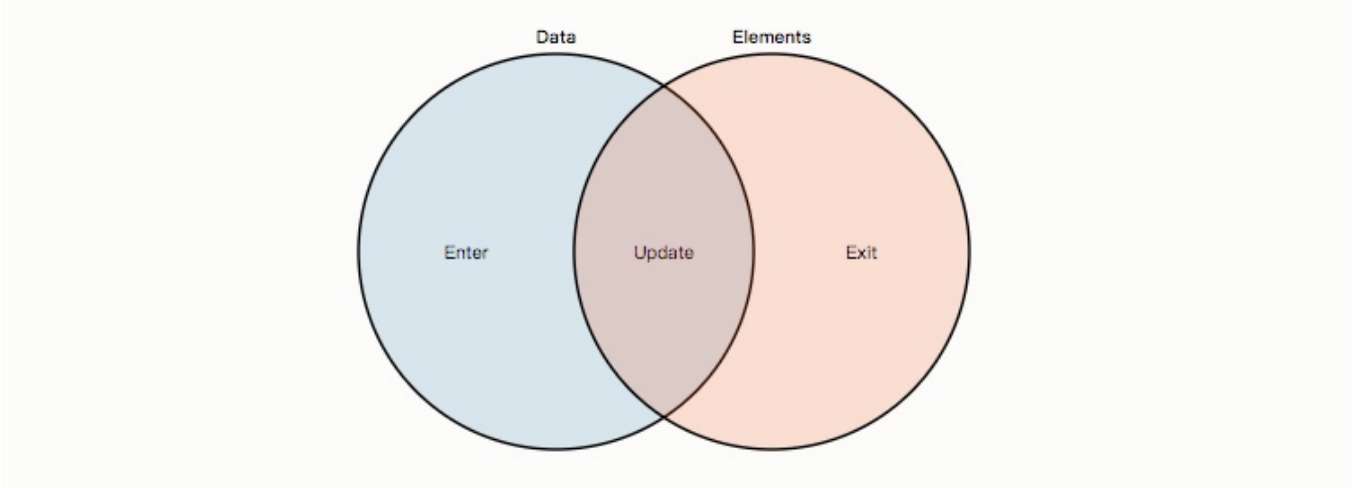
```
svg.append('circle')
    .attr("cx", d.x)
    .attr("cy", d.y)
    .attr("r", 2.5)
```

但这样每次只能创建一个`circle`元素，然而我们需要一堆圆来传达数据内容。在你想要使用`for`循环或者暴力解决这个问题前，请理解下面例子中的链式方法的意义。

```
svg.selectAll("circle")
  .data(data)
  .enter().append("circle")
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; })
    .attr("r", 2.5);
```

上面的代码做了我们想做的：为每个数据元素创建了对应的`circle`元素，赋值`x`、`y`属性定位到对应位置。在上面的代码中`selectAll('circle')`做了什么？为什么我们需要选择不存在的元素来创建新的元素？

这是约定俗成的写法，与其告诉D3如何来做事，不如直接说你想要什么。我们想要`circle`元素来对应数据信息，每个圆对应一份数据元素。为此我们告诉D3我们需要一个与数据对应的`cirlce`元素选择集，而不是要求D3创建`circle`元素。这个思想就是我们本次所要说明的**数据绑定**。



当数据绑定到已存在元素上时，会创建`update`选择集。其余未被绑定到元素上的数据会创建`enter`选择集，表示缺失的元素。同样，已存在且未被绑定到数据的元素会返回`exit`选择集，通常表示需要被删除的元素。

下面我们通过解释代码来一步步说明`enter-append`模式数据绑定的意义：

1. 首先，`svg.selectAll('circle')`返回一个全新的空选择集，因为SVG容器中是空的，这个选择集的父元

素节点就是SVG容器本身。
2. 接着这个选择集与一个数组进行数据绑定，返回三个表示三种状态的新选择集：`enter`、`update`、`exit`。因为这些选择集现在是空的，所以返回的`update`、`exit`也是空的。但是因为已经绑定数组了，所以此时`enter`选择集包含对应每个数据元素的占位符。
3. 当执行`selection.data`方法时会返回`update`选择集，`enter`和`exit`选择集不会返回。执行`selection.enter`则会返回对应的`enter`选择集。
4. 在`enter`选择集上执行`selection.append`方法会在SVG容器中添加缺失的元素。现在在SVG容器中，每个数据项都有一个对应的`circle`元素存在了。

数据绑定的本质是先声明选择集和数据之间的关系，然后通过`enter`、`update`和`exit`三个状态来具体实现这个关系。

为什么这么麻烦？就不能提供一个直接创建多元素的原生方法吗？数据绑定的好处在于它可以泛化。当以上代码只是处理了`enter`选择集，这对静态可视化来说已经足够了。你也可以扩展代码让它支持动态图表，只需要稍微修改下`update`和`exit`选择集的操作。这意味着你可以可视化实时数据，允许交互探索，并可以平稳过度数据的变化。

下面是一个小例子：

```
var circle = svg.selectAll("circle")
  .data(data);

circle.exit().remove();

circle.enter().append("circle")
    .attr("r", 2.5)
  .merge(circle)
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; });
```

每当运行此代码时，它将重新计算数据绑定并维护元素和数据之间所需的对应关系。如果此时新数据集数量少于老数据集时，剩余的元素将出现在`exit`选择集当中并且会被删除。如果新数据集中元素数量多于老数据集时，`enter`选择集会为这些数据生成占位符，并创建对应的`circle`元素。如果新数据集和老数据集数量相同，则现有元素会更新到新数据集对应的位置，没有元素增减。

以数据绑定的思想来编写意味着你的代码更具有声明性：你可以处理这三种状态无需任何条件语句或循环。相反，你可以描述元素应该如何与数据相对应。如果返回的`enter`、`update`、`exit`选择集为空，对应的代码是不会执行的。

如果有需要，数据绑定还允许你操作元素为特定状态。比如你可以设定给`enter`选择集的元素设定属性（例如圆半径，定义`r`属性）。

```
circle.enter().append("circle")
    .attr("r", 0)
  .transition()
    .attr("r", 2.5);
```

后记

本文翻译自D3作者Mike Bostock2012年的博文，翻译本文的目的是希望可以从作者本身的角度来理解为什么要用绑定数据和元素的方式来完成创建多SVG元素的操作。

原文链接