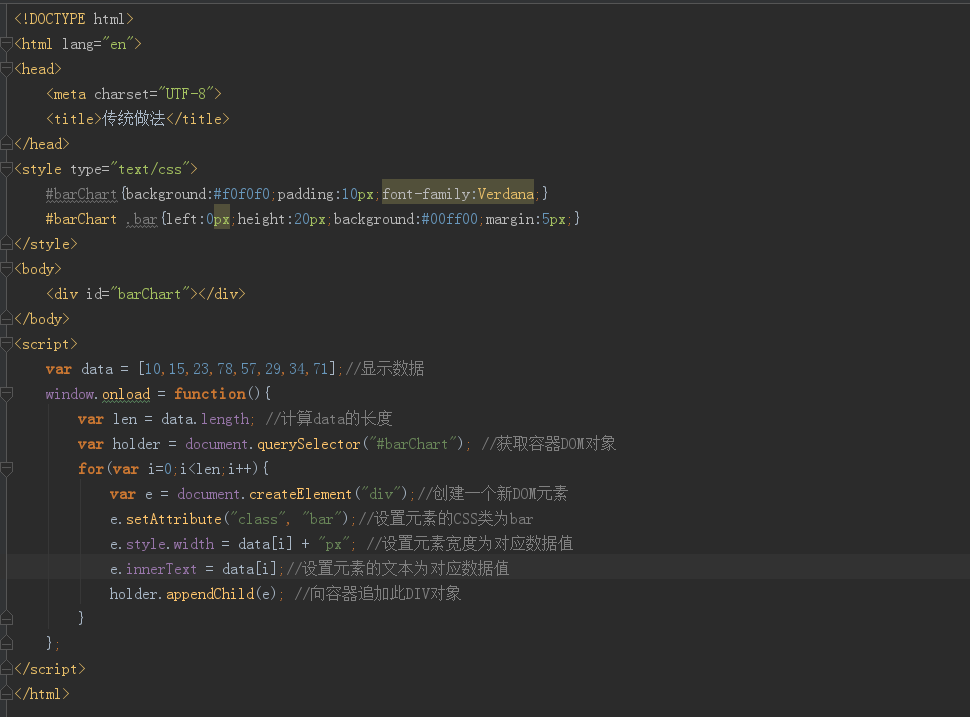
D3.JS（data-drive documents）数据驱动文档

1.传统的做法



2.数据可视化：将数据以图形的方式表现出来的过程就叫做数据可视化。

3. d3.js是一个数据可视化库；

* d3.js不是一个图形绘制库

d3依赖于标准的web技术来绘制可视化元素，比如 HTML、SVG、CSS。使用d3需要我们对这些标准技术有基础的了解。

* d3.js是一个基于集合概念的DOM操作库

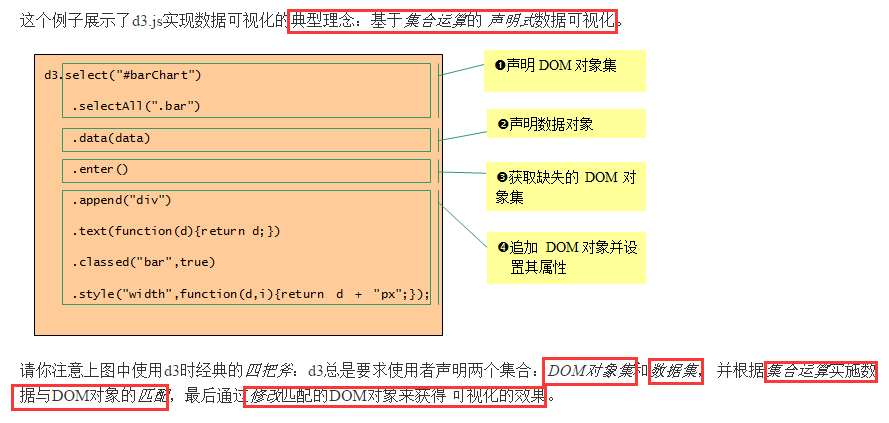
d3不是图形绘制库，但它对DOM操作进行了封装。和jQuery类似，d3依赖于选择符选 中一组元素，建立一个集合，然后使用集合对象的方法操作DOM。

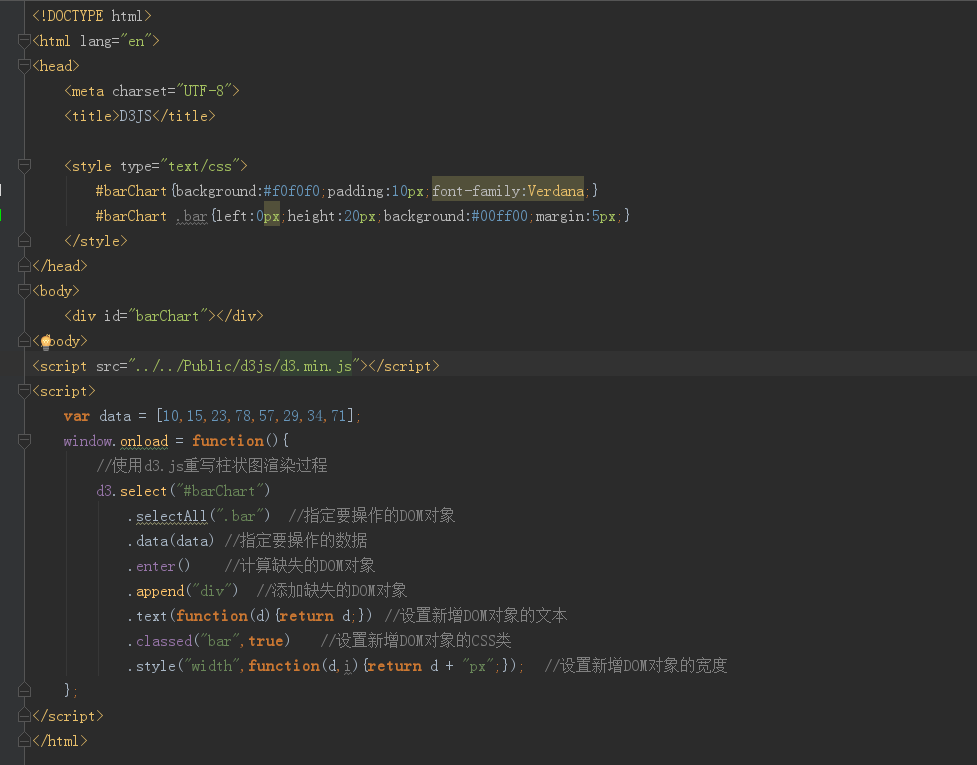
* d3.js的大量功能集中在数据处理方面

要将数据映射到图形，有很多琐碎的工作，比如数据范围的变换、插值的计算、布局的 计算等等。d3.js的大量功能是集中在这样的数据处理方面的。

* d3.js的核心是对数据和可视化元素的匹配

d3将数据可视化抽象为数据与可视化元素的匹配，一个数据对应一个可视化元素，一个 数值对应一个可视化元素的属性。d3封装了这个匹配的复杂过程，让我们得以简单的 通过声明数据和可视化元素来完成数据可视化的任务。

4. 



重点：

* 1. 声明DOM对象集：d3的*数据可视化流程*总是从选中一组DOM元素建立一个*集合对象*开始



试图在文档DOM树中选择div#barChart中的所有div.bar。 我们将这一步返回的DOM集合保存到变量selection1中。

你应该会注意到，这是一个空集合！

这有点意思。

* d3允许声明一个空集合

在jQuery中，如果我们的选择符没有选中任何HTML元素，那么后面的所有操作都没有意义了。 但是，d3允许我们选不中任何元素来来建立一个空集合。

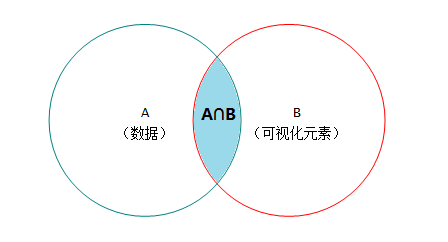
因为，d3还有第2斧，用数据来影响这个空集合。

size()方法返回集合中所有DOM元素的数量。

2.声明数据集

**匹配计算**

*data()*方法执行时对数据集和DOM元素集进行了*匹配*计算， 直接返回的结果就是两个集合的*共有部分*。由于DOM元素集是空的，所以结果一定也没有任何DOM元素：



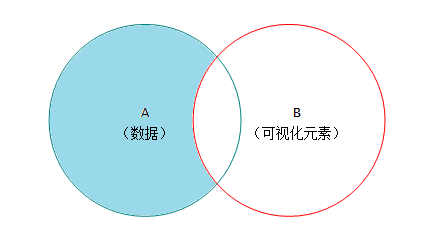
但是，匹配计算的结果有两个重要的方法，让我们可以获得数据集合DOM元素集的差异在哪里：

* enter() : 获得数据集中比DOM元素集中多出来的数据
* exit() ：获得DOM元素集中比数据集中多出来的数据

3.获得结果集



由于原始DOM集合为空，而数据集有8项，那么selection3中也将有8项描述8个缺失的 DOM对象：



4.实施DOM操作

（1）使用集合的append()方法，我们创建缺失的DOM对象：



（2）使用text() 方法设置每个元素的文本内容：



注意到text()方法的参数是一个函数，selection4是一个集合，内有8个div元素，我们经过之前的三个步骤，这每个元素都已经有对应的数据了（d3负责管理其对应关系），比如，第一个DIV元素对应的是10，第二个DIV元素对应 的是15......

每当集合的方法被调用时，它都检查传入的参数，如果传入的参数是一个函数，d3就对集 合中的每个DOM元素执行一次这个函数，并传入这个DOM元素对应的数据。

（3）使用classed()方法设置每个元素的CSS类为"bar"：



（4）使用width()方法使用每个元素对应的数据，设置其宽度

