

- 本文档是D3官方文档中文翻译，并保持与[最新版](#)同步。
- 如发现翻译不当或有其他问题可以通过以下方式联系译者：
- 邮箱：zhang_tianxu@sina.com
- QQ群：[D3.js](#):437278817, [大数据可视化](#): 436442115
- Github小组：[VisualCrew](#): <https://github.com/VisualCrew>
- 本页译文，我们作如下约定：
- **domain**：译为 输入域（或者定义域）
- **range**：译为 输出范围（或值域）
- **band**：译为 区间段（或频段宽度），可以理解为线段
- **参数**：也就是 function 的参数，译为 入参，比如：`function abc(x, y) {}`，函数 `abc` 的两个参数 `x` 和 `y`，我们称为：入参 `x`、入参 `y`

比例尺是一系列函数，用来映射输入域到输出范围。**序数比例尺**的输入域是离散的，比如：一组名称或类别。还有[[定量比例尺|数值比例尺]]，其输入域是连续的，比如：实数集、日期。比例尺在D3中是一个可选的功能，如果你喜欢自己处理数学的话大可不必使用它们。然而使用比例尺可以大大简化映射数据到可视化编码的必需代码量。

一个比例尺对象，例如：[d3.scale.linear](#) 的返回值既是一个对象也是一个函数。因此，可以像函数一样使用比例尺对象。并且比例尺有额外的函数可以改变它的行为。就像 D3 中的其他类，比例尺同样支持链式语法 `setter` 方法直接返回比例尺对象，允许多个 `setter` 方法在一个简洁的语句中调用。

```
# d3.scale.ordinal()
```

构造一个新的序数比例尺，使用空的输入域和输出范围。如果序数比例尺没有指定输出范围，取值时总会返回 `undefined`。

```
# ordinal(x)
```

传入一个输入域中的值 `x`，返回对应输出范围中的值。如果输出范围已指定（比如通过 [range](#) 指定，但是不是通过 [rangeBands](#)、[rangeRoundBands](#) 或 [rangePoints](#) 来指定的），并且入参 `x` 的值不在输入域中，那么 `x` 就会被隐含地添加进到输入域中；这之后，当使用相同的值 `x` 再次调用该函数时就会返回输出范围中相同的值 `y`。

```
# ordinal.domain([values])
```

获取或指定当前比例尺对象的输入域。

如果指定了入参 `values` 的值，就会设置当前比例尺对象的输入域为指定的 `values` 数组。`values` 数组中的第一个元素会被映射到输出范围中的第一个值、第二个元素会被映射到输出范围中的第二个值等等。输入域 `values` 在内部会被存储到一个关联数组中作为从值到索引的映射，生成的索引会被用来从输出范围中取值，也就是取对应索引的值；这样的话，序数比例尺的输入域中的值就会被强制转换为字符串，并且唯一地标识到输出范围中对应的值。

如果没有指定 `values`，这个方法就会返回当前的域。

序数比例尺的输入域是否指定是可选的，但是输出范围是必须明确地指定。因为，如 [ordinal\(x\)](#) 所述，如果 `x` 的值在输入域中不存在，就会被隐式的新增到输入域中；换句话说，输入域是从使用情况中推断而来的。虽然输入域可能被隐式构造，最好还是显式指定序数比例尺的域，以保证确定性的行为，而从使用情况来推断输入域是取决于顺序。

```
# ordinal.range([values])
```

获取或指定当前比例尺对象的输出范围。

如果指定了入参 `values` 的值，就会设置当前比例尺对象的输出范围为指定的 `values` 数组。输入域中的第一个元素会被映射到 `values` 的第一个值、输入域中的第二个元素会被映射到 `values` 的第二个值等等。如果 `values` 中的值的个数少于输入域中元素的个数，那么 `values` 中的值会被循环使用。

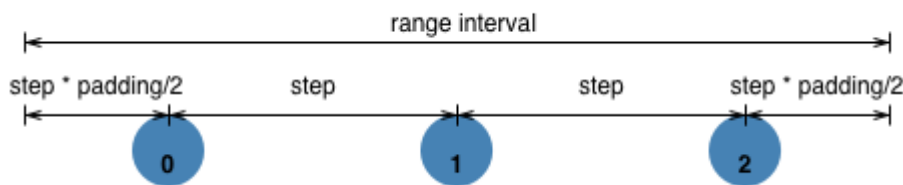
如果没有指定 *values*，这个方法就会返回当前的域。

这个函数也可以被用来指定一组已经被明确计算好了的离散的输出范围，比如：一组类目型的颜色。其他一些情况，比如：序列散点图、柱状图使用 [rangePoints](#) 或 [rangeBands](#) 会更为方便，具体请先参考对应图表的案例代码。

ordinal.[rangePoints](#)(interval[, padding])

功能同[ordinal.range\(values\)](#)，只是适用范围不同。

指定输出范围为一个连续的区间 *interval*；*interval* 需要两个数值元素，第一个表示区间的最小值、第二个表示区间的最大值。区间 *interval* 会被细分为 *n* 个等间隔的刻度“点”，*n* 的大小取决于输入域数组的真实长度（也就是数组中每个元素的唯一性而确定的长度）。在这些被细分的刻度点中，第一个点的起始位置和最后一个点的结束位置会因为入参 *padding* 的值而做相应消减（见下图），消减长度是：*padding* 个间隔长度的一半；默认情况下 *padding* 是 0。*padding* 的值会当做间隔的倍数来使用。



```
var o = d3.scale.ordinal()
    .domain([1, 2, 3, 4])
    .rangePoints([0, 100]);

o.range(); // [0, 33.333333333333336, 66.66666666666667, 100]

o.rangePoints([0, 120], 1);
o.range() // [15, 45, 75, 105]    前面被空了 30*1/2=15 后面也被空了 15; 其中 30 是间隔宽度

o.rangePoints([0, 120], 2);
o.range() // [24, 48, 72, 96]    前面被空了 24*2/2=24 后面也被空了 24; 其中 24 是间隔宽度

o.rangePoints([0, 120], 3);
o.range() // [30, 50, 70, 90]    前面被空了 20*3/2=30 后面也被空了 30; 其中 20 是间隔宽度
```

ordinal.[rangeRoundPoints](#)(interval[, padding])

功能同[ordinal.rangePoints\(interval\[, padding\]\)](#)，但是该函数可以美化输出的刻度点，也就是保证整数。

```
var o = d3.scale.ordinal()
    .domain([1, 2, 3, 4])
    .rangeRoundPoints([0, 100]);

o.range(); // [1, 34, 67, 100]
```

需要提及的，凑整肯定会导致额外的 *padding* 被增减，通常是和输入域的长度成一定比例；修改输出范围的区间长度，使其更紧凑便可以大大减少额外的 *padding* 被使用。

```
var o = d3.scale.ordinal()  
  .domain(d3.range(50))  
  .rangeRoundPoints([0, 95]);  
  
o.range(); // [23, 24, 25, ..., 70, 71, 72]  
o.rangeRoundPoints([0, 100]);  
o.range(); // [1, 3, 5, ..., 95, 97, 98]
```

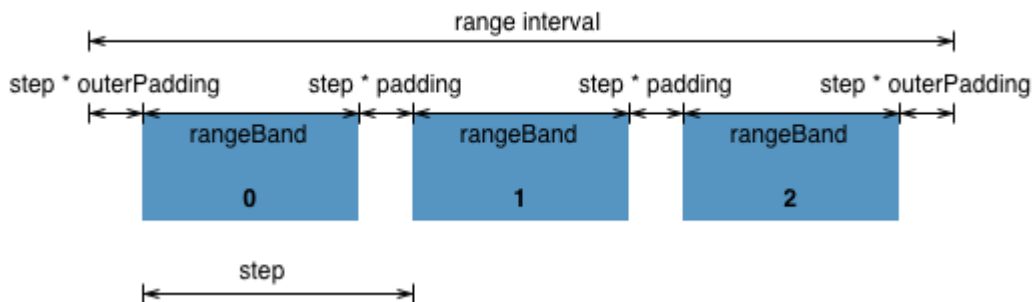
(或者，你也可以手动的处理，这真的有必要吗???)

`ordinal.rangeBands(interval[, padding[, outerPadding]])`

功能同[ordinal.rangePoints\(interval\[, padding\]\)](#)，但是该函数是将区间切分成一个个小的区间段，而不是一个个刻度“点”。

指定输出范围为一个连续的区间 *interval*；*interval* 需要两个数值元素，第一个表示区间的最小值、第二个表示区间的最大值。区间 *interval* 会被切分为 *n* 个等间隔区间段，*n* 的大小取决于输入域数组的真实长度（也就是数组中每个元素的唯一性而确定的长度）。每个区间段的宽度会因为打区间的首尾 *outerPadding* 值和每个区间段的 *padding* 值而有所消减，默认情况下 *padding* 是 0。通常，*padding* 的取值范围是 $[0, 1]$ ，表示相邻区间段间的间隔（或空白）占区间段的比例；比如：`padding=0.5` 表示区间段的实际宽度与相邻区间段间的留白相等，参考下面图片的说明。

outerPadding 表示第一个区间的起始位置和最后一个区间的结束位置的留白，留白的长度与 *padding* 的使用方式类似，`outerPadding=0` 表示首尾顶着边缘。



```
var o = d3.scale.ordinal()  
  .domain([1, 2, 3])  
  .rangeBands([0, 100]);  
  
o.rangeBand(); // 33.333333333333336  
o.range(); // [0, 33.333333333333336, 66.66666666666667]  
o.rangeExtent(); // [0, 100]
```

`ordinal.rangeRoundBands(interval[, padding[, outerPadding]])`

功能同[rangeBands](#)，但是该函数可以美化输出的区间段，也就是保证每个区间段的起点值都是整数。

```
var o = d3.scale.ordinal()  
  .domain([1, 2, 3])  
  .rangeRoundBands([0, 100]);  
  
o.range(); // [1, 34, 67]  
o.rangeBand(); // 33  
o.rangeExtent(); // [0, 100]
```

需要提及的，凑整肯定会导致额外的 *padding* 被增减，通常是和输入域的长度成一定比例；修改输出范围的区间长度，使其更紧凑便可以大大减少额外的 *padding* 被使用。

```
var o = d3.scale.ordinal()  
  .domain(d3.range(50))  
  .rangeRoundBands([0, 95]);  
  
o.range(); // [23, 24, 25, ..., 70, 71, 72]  
  
o.rangeRoundBands([0, 100]);  
o.range(); // [0, 2, 4, ..., 94, 96, 98]
```

(或者，你也可以手动的处理，这真的有必要吗???)

[# ordinal.rangeBand\(\)](#)

获取区间段的宽度。只有当使用 [rangeBands](#) 或 [rangeRoundBands](#) 来指定输出范围时才有效，否则返回一律返回 0。

[# ordinal.rangeExtent\(\)](#)

获取当前比例尺对象的未被切分的输出范围：一个两元素的数组，第一个元素表示最小值、第二个元素表示最大值。


[# ordinal.copy\(\)](#)


深度拷贝一个当前比例尺对象的副本，并返回，更改这个副本的属性不会影响到原比例尺对象的属性。


Categorical Colors


[# d3.scale.category10\(\)](#)


构造一个新的序数比例尺，使用以下10种类型的颜色：


 1f77b4 #1f77b4


 ff7f0e #ff7f0e

 2ca02c #2ca02c


 d62728 #d62728

 9467bd #9467bd

 8c564b #8c564b

 e377c2 #e377c2


 7f7f7f #7f7f7f


 bcbd22 #bcbd22


 17becf #17becf


<#> d3.scale.**category20**()


构造一个新的序数比例尺，使用以下20种类型的颜色：


 1f77b4 #1f77b4

 aec7e8 #aec7e8


 ff7f0e #ff7f0e


 ffbb78 #ffbb78


 2ca02c #2ca02c


 98df8a #98df8a


 d62728 #d62728


 ff9896 #ff9896


 9467bd #9467bd


 c5b0d5 #c5b0d5


 8c564b #8c564b


 c49c94 #c49c94


 e377c2 #e377c2

 f7b6d2 #f7b6d2


 7f7f7f #7f7f7f

 c7c7c7 #c7c7c7

 bcbd22 #bcbd22

 dbdb8d #dbdb8d


 17becf #17becf

 9edae5 #9edae5


[# d3.scale.category20b\(\)](#)


构造一个新的序数比例尺，使用以下20种类型的颜色：

 393b79 #393b79

 5254a3 #5254a3


 6b6ecf #6b6ecf

 9c9ede #9c9ede


 637939 #637939


 8ca252 #8ca252

 b5cf6b #b5cf6b


 cedb9c #cedb9c


 8c6d31 #8c6d31


 bd9e39 #bd9e39

 e7ba52 #e7ba52

 e7cb94 #e7cb94

 843c39 #843c39


 ad494a #ad494a

 d6616b #d6616b

 e7969c #e7969c

 7b4173 #7b4173


 a55194 #a55194

 ce6dbd #ce6dbd


 de9ed6 #de9ed6


[# d3.scale.category20c\(\)](#)

构造一个新的序数比例尺，使用以下20种类型的颜色：


 3182bd #3182bd

 6baed6 #6baed6


 9ecae1 #9ecae1


 c6dbef #c6dbef

 e6550d #e6550d


 fd8d3c #fd8d3c


 fdae6b #fdae6b


 fdd0a2 #fdd0a2


 31a354 #31a354

 74c476 #74c476

 a1d99b #a1d99b


 c7e9c0 #c7e9c0


 756bb1 #756bb1


 9e9ac8 #9e9ac8


 bcbddc #bcbddc

dadaeb #dadaeb

636363 #636363

969696 #969696

bdbdbd #bdbdbd

d9d9d9 #d9d9d9

ColorBrewer

D3也通过 [[Cynthia Brewer|<http://colorbrewer2.org/>]] 绑定了一些奇妙的类目颜色比例尺。可以从 [lib/colorbrewer](#) 找到这些颜色比例尺的 CSS 或 Javascript 实现。

对于 CSS，只要在需要着色的元素上指定 class 类如：像"q0-3"、"q1-3"或"q2-3"，然后，在父元素（例如SVG元素）上设置需要的颜色比例尺的名称到 class 类属性如"RdBu"或"Blues"。具体参见：[calendar heatmap](#)、[choropleth](#)

对于 JavaScript，可以使用 colorbrewer.RdBu[9] 或等同的方法作为 **d3.scale.ordinal** 的范围如：

```
var o = d3.scale.ordinal()  
    .domain(["foo", "bar", "baz"])  
    .range(colorbrewer.RdBu[9]);
```

	人员	组织	时间
翻译	大傻	VisualCrew/小组	2014-11-24 23:21:05
校对	WeiFei365	VisualCrew/小组	2015-12-01 22:32:09
排版	WeiFei365	VisualCrew/小组	2015-12-01 15:08:58

- 目前，本页结构完全和英文页相同
- 已同步更新首页的链接；
- 其他页面的引用可以像英文文档那样直接引用；