
- 本文档由VisualCrew小组耗时两年翻译，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
- API使用方法可参考：https://github.com/tianxuzhang/d3-api-demo

D3的**时间比例尺(Time scale)**是d3.scale.linear比例尺的扩展，使用Javascript的Date对象作为其输入域(domain)。因此，不同于普通的线性比例尺（linear scale），domain的值会被强制转为时间类型而非数字类型；同样的，反函数(invert)返回的是一个时间类型。最方便的是，time scale同样提供了基于time intervals的合适的打点器(ticks)，解除了为任何基于时间的域生成轴的痛苦。

d3.time.scale返回的是一个比例尺对象，同时也是一个函数。你可以像任何函数一样引用它，同时它有额外的方法可以改变它的行为。如D3的其它类一样，scale对象也可以链式调用方法，setter方法返回scale对象本身，可以用一个简洁的声明中引用多个setters。

d3.time.**scale**()

使用默认的域和范围来构建一个新的时间比例尺；默认以本地时间配置ticks和tick format。

d3.time.scale.**utc**()

使用默认的域和范围来构建一个新的时间比例尺；默认以UTC时间配置ticks和tick format。

scale(*x*)

给定输入域内的时间*x*，返回相应输出范围的值；

scale.**invert**(*y*)

对输出范围*y*内的值，返回相应的输入域*x*内的时间对象。这个用于从rang到domain的反向映射。对于输出域range内的*y*值，scale(scale.invert(*y*)) 等于 *y*；同样，对于输入域domain内的时间对象*x*，scale.invert(scale(*x*)) 等于 *x*。这种反向操作符在交互时非常有用，例如，鼠标移动时，计算出鼠标当前位置相对应的输入域的时间。

scale.**domain**([*dates*])

如果设置了参数*dates* ，则把scale的输入域domain设置为dates数组，该数组必须包含两个及以上的时间对象。如果数组里的元素不是时间对象，将被强制转换为时间对象；这个强制转换在比例尺被引用时发生。如果没有设置参数*dates*，则返回scale的当前的输入域domain。通常来说，时间比例尺虽然在输入域domain里只有两个时间对象，但你可以为*polylinear* 比例尺设置多于两个的时间对象，这样的话，必须在输出范围中设置相同数量的数值。

scale.**nice**([*interval*[, *step*]])
scale.**nice**([*count*])

扩展域使得开始和结束于很好的整数值，当由指定的时间间隔(time *interval*)和可选参数阶数*step*。作为替代方案，指定一个明确的时间间隔，一个数字*count* 可以被指定，并且在时间间隔将自动被选择为与scale.ticks一致。如果未指定计数count ，则默认为10。

这个方法通常扩展比例尺的域，可能只扩展边界到最接近的整数值。Nicing是有用的，如果域是从数据计算的并可以是不规则的。例如，对于域[2009-07-13T00:02, 2009-07-13T23:48]优化域是[2009-07-13, 2009-07-14]。如果域有多于两个值，优化将只影响域的第一个和最后一个值。

scale.**range**([*values*])

如果设置了参数values，则把scale的输出域range设置为values数组，该数组必须包含两及以上的数值，以匹配输入域domain。values数组里的元素不需要一定是数字类型，任何支持下面的interpolator的数据类型都可以。但是，invert操作符需要数字类型的输出域。如果没有设置参数values，则返回scale的当前输出域。

scale.**rangeRound**([*values*])

设置scale的输出域range为指定的values数组，同时设置了scale的interpolator为d3.interpolateRound。如果比例尺输出的值应该是整数时，这是一个便捷的方法，可以有效地避免平滑处理工件。也可以在引用比例尺应用之后手动四舍五入输出的值。

scale.**interpolate**([*factory*])

如果设置了参数factory，则将scale的输出插值器设置为factory。默认的插值器工厂为d3.interpolate，将标准化的[0,1]范围内的输入域参数t，映射到相应的输出域range内的数值。插值器工厂将为输出范围内的相邻的每对数值构造插值器。

scale.**clamp**([*boolean*])

如果设置了参数boolean（布尔值），则相应地开启或者关闭clamping。默认情况下，clamping是关闭的，如果一个超出了输入域的值被传递给了比例尺，则比例尺将通过线性推断，可能会返回一个输出范围之外的数值。以默认的[0,1]输入域和输出域为例，输入数值2将返回输出数值2。如果clamping开启，标准化的输入域参数t会被限定在输出域[0,1]内，这样比例尺返回的值永远都在输出域内。如果boolean没有设置，返回的是比例尺是否限定当前数值到输出范围内。

scale.**ticks**([*interval*[, *step*]])
scale.**ticks**([*count*])

返回比例尺输入域的代表性时间。返回的打点tick时间，它们有相同间距（模式不标准的时间间隔，比如月份和闰年）、包含可读性的数值（比如午夜），以及确保在输入域范围之内。刻度常被用在显示参考线、刻度标记，与视觉化的数据一起使用时。

如果参数*count*是一个数字，将返回大约*count*数量的刻度。如果未设置参数*count*，默认设置为10。*count*只是一个提示，根据输入域domain，比例尺可能会返回更多或更少的数值。如果设置了参数时间间隔(time *interval*)，那么将会引用time interval的值域函数(range function)，同时传递可选参数*step*，以生成刻度。比如，生成默认的10个刻度:

```
scale.ticks(10);
```

或者，以15分钟为间隔生成ticks：

```
scale.ticks(d3.time.minute, 15);
```

注意：对于UTC比例尺，使用相应的UTC输出范围方法（比如，`d3.time.minute.utc`）。

以下时间间隔被视为自动ticks：

- 1-, 5-, 15- 和 30-second.
- 1-, 5-, 15- 和 30-minute.
- 1-, 3-, 6- 和 12-hour.
- 1- 和 2-day.
- 1-week.
- 1- 和 3-month.
- 1-year.

这一组时间间隔是稍微有点随意的，并且额外的值也会在未来会被加上。

scale.**tickFormat**()

返回一个可以显示刻度值的世时间格式化函数。参数count应该和用于生成刻度值的数量一致。你不用必须使用比例尺内建的刻度格式，但它会根据输入域的时间自动计算相应的结果。 比如以下时间格式：

- %Y – 年分界，如"2011".
- %B – 月分界，如"February".
- %b %d – 星期分界，如"Feb 06".
- %a %d – 日期分界，如"Mon 07".
- %I %p – 小时分界，如"01 AM".
- %I:%M – 分钟分界，如"01:23".
- :%S – 秒分界，如":45".
- .%L – 毫秒分界，如".012".

使用多重时间格式，默认的时间格式为每个时间间隔提供局部和全局环境。例如，为显示[11 PM, Mon 07, 01 AM]，刻度格式器可以同时展现小时和日期信息--而非只有小时。如果你喜欢用单一的时间格式器(d3.time.format)，你可以一直使用你自己定义的多比例尺时间格式(custom multi-scale time format)。

scale.**copy**()

返回当前time scale的完整副本。所有对当前世家比例尺的改变都不会改变返回的副本，反之亦然。

本文参与	人员	组织	时间
翻译	何凯琳	VisualCrew小组	20141124
校对/排版	大傻	VisualCrew小组	20141129
	liang42hao	VisualCrew小组	2016-04-05 13:44:32