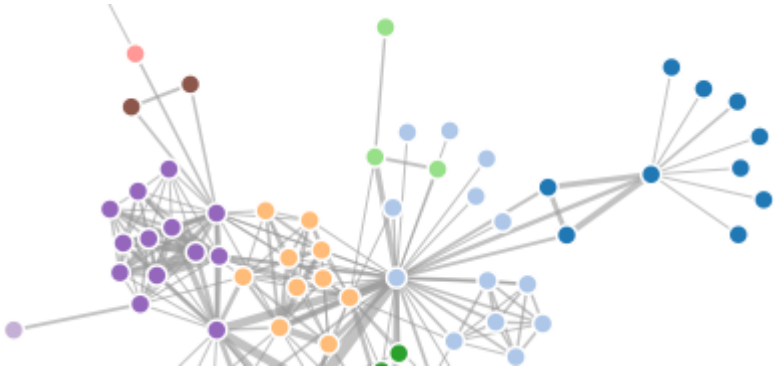
> *Wiki* ‣ *[[API--中文手册]]* ‣ *[[布局]]* ‣ ***力布局***

- 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数据可视化205076374，大数据可视化436442115

一个灵活的力导向图布局实现使用位置Verlet整合算法(Verlet integration)允许简单地控制(simple constraints)。有关物理模拟的更多信息，请参见Thomas Jakobsen。这个实例是利用四叉树(quadtree)加速电荷相互作用使用Barnes–Hut近似(Barnes–Hut approximation)。除了电荷力(charge force)，伪重力(pseudo-gravity)保持节点中心在可见区域并且避免排出断开子图，当链接的固定距离(fixed-distance)的几何约束时。额外的自定义力和约束可应用于"tick" 事件,简单地通过更新节点的*x*和*y*属性。



详见: 此贴 和 corresponding deck.

一些例子：

- divergent forces
- multiple foci
- graph constructor
- force-directed tree
- force-directed symbols
- force-directed images and labels
- force-directed states
- sticky force layout

像D3的其他类，布局遵循方法链模式setter方法返回布局自身。允许使用一个简单的声明调用多个setter方法。有别于其他的布局实现是无状态的。力导向图布局 内部保持关联节点和链接的引用。因此，一个给定的力导向图布局实例只可以和一个单一的数据集一起使用。

d3.layout.**force**()

构造一个新的力导向布局使用默认设置：尺寸 1×1，链接长度1，摩擦0.9，距离20，充电强度-30，重力强度0.1和θ参数0.8。默认的节点和链接是空数组，并且当布局开始时，内部的α冷却参数被设置成0.1。构建力导向图布局的通用模式是所有配置属性, 然后调用开始start函数：

```
var force = d3.layout.force()
    .nodes(nodes)
    .links(links)
    .size([w, h])
    .linkStrength(0.1)
    .friction(0.9)
```

```
    .linkDistance(20)
    .charge(-30)
    .gravity(0.1)
    .theta(0.8)
    .alpha(0.1)
    .start();
```

需要注意的是, 像D3的其他布局，力导向图布局不要求特定的可视化表现。最常见的，节点被映射到SVG圆形元素，链接被映射为SVG线元素。但你也可以显示节点作为符号(symbols)或图片(images)。

force.**size**([*width, height*])

如果指定了*size*，设置可用的布局大小为指定的代表*x*和*y*的两元素数字数组来。如果未指定*size*，返回当前size，默认为 `[1, 1]` 。size影响力导向图的两个方面：重力中心和初始的随机位置。重心是 [*x* / 2, *y* / 2]。当节点被添加到力导向图布局，如果不具有已设置的*x*和*y*属性，然后这些属性都分别使用范围为[0, *x*]和[0, *y*]的均匀随机分布进行初始化。

force.**linkDistance**([*distance*])

如果指定了*distance* ，设定链接节点间的目标距离为指定的值。如果未指定*distance* ，返回布局的当前链路距离，默认为20。如果*distance* 是常量，那么所有的链接是相同的距离。否则，如果*distance*是一个函数，则该函数为每个链接（按顺序）求值，被传递参数是链接和它的索引，用 `this` 上下文作为力导向图布局；该函数的返回值被用来设置每个链接距离。当布局开始(starts)时该函数被求值。通常, 距离被指定以像素为单位; 然而，这些单位也可以是相对于布局的size.

链接不实现为在一般力导向图布局中的"弹性力"，而是弱几何约束。对于布局的每一个tick，计算每对链接节点之间的距离并与目标距离进行比较；链接随后朝向彼此移动或远离彼此，以收敛到所需的距离。这种在Verlet 集成算法之上约束松弛的方法大大稳定于之前使用弹性力的方法，并且还允许在Tick事件监听器灵活实现其他约束(other constraints)，如层次分层(hierarchical layering)。

force.**linkStrength**([*strength*])

如果指定了*strength* ，设置链接间强度（刚性）为[0,1]范围内的指定的值。如果*strength* 未被指定，返回布局的当前链接强度，默认为1。如果强度*strength*是一个常数，那么各个链接都有相同的强度。否则，如果强度是一个函数，那么该函数为每一个链接（按顺序）求值，传递的参数是链接和它的索引，this 上下文是这个力布局；该函数的返回值随后被用来设置每个链接的强度。每当布局开始(starts)时调用这个函数。

force.**friction**([*friction*])

如果指定了摩擦系数*friction* ，设定摩擦系数为指定的值。如果摩擦系数未被指定，返回当前系数，默认为0.9。这个参数的名称可能有误导性的；它不对应标准物理的摩擦系数(coefficient of friction)。相反，它更接近速度衰减：在模拟的每个tick，粒子速度通过指定的*friction*缩减。因此，值1对应于无摩擦的环境中，而一个0值冻结所有颗粒就位。不推荐超出范围[0,1]的值，可能有破坏稳定的的影响。

force.**charge**([*charge*])

如果指定电荷强度*charge* ，设置电荷强度为指定的值。如果电荷强度*charge* 未被指定，返回电流电荷强度，其默认值为-30。如果电荷强是常量，那么所有节点都具有相同的电荷。否则，如果电荷强度*charge*是一个函数，则该函数为每个节点（按顺序）求值，传递的参数是节点和它的索引， `this` 上下文作为力布局；该函数的返回值被用于设置每个节点的电荷强度。每当布局开始(starts)时被调用。

负值导致节点排斥，而正值导致节点吸引。对于图形布局，应使用负值；对于N体模拟(*n*-body simulation)，可以使用正值。所有节点都假定为无穷远的小点具有相等电荷和质量。电荷力是通过Barnes–Hut算法(Barnes–Hut algorithm)高效实现的，为每个tick计算四叉树。电荷力设置为零禁用四叉树(quadtree)，它可以显着提高性能，如果你不需要N体模拟。 Barnes–Hut算法：http://arborjs.org/docs/barnes-hut

force.**chargeDistance**([*distance*])

如果*distance* 被指定，设置电荷强度已经应用的最大距离。如果*distance* 未被指定，返回当前最大电荷距离，默认为无穷大。指定一个有限电荷距离提高力导向图的性能和产生更本地化的布局。限定距离的电荷力是尤其有用当合自定义重力一起使用时。有关示例, 请参阅"限定距离的电荷力(Constellations of Directors and their Stars)" (*The New York Times*).

force.**theta**([*theta*])

If *theta* is specified, sets the Barnes–Hut approximation criterion to the specified value. If *theta* is not specified, returns the current value, which defaults to 0.8. Unlike links, which only affect two linked nodes, the charge force is global: every node affects every other node, even if they are on disconnected subgraphs.

若指定了 *theta*, 将其设定为Barnes–Hut近似的判定参数. 若 *theta* 未指定, 返回当前值, 默认为 0.8. 与其他只影响两个节点的连接不同, 此电荷力是全局的: 所有节点相互影响, 甚至是在未连通的子图里.

为了避免大图的二次性能下降。力导向图布局使用Barnes–Hut近似(Barnes–Hut approximation)，每个tick花费O(*n* log *n*)。对于每一个tick，创建一个四叉树用于存储当前节点的位置；然后，对每个节点，计算给定节点的所有其他节点的总电荷力。为了聚集过远的节点，通过处理节点的距离的簇作为单个逼近电荷力。*Theta* 确定计算精度：对一个节点到块象限中心的距离，四叉树中象限区域的比率小于*theta*，在给定象限的所有节点被视为一个单一的，大的节点，而不是单独地计算。

force.**gravity**([*gravity*])

如果指定了重力*gravity* ，设置引力强度为指定的值。如果未指定重力，返回当前的引力强度，默认为0.1。这个参数的名称可能是误导性的：它不对应于物理重力gravity（可以用一个正电荷参数(charge)进行仿真)。相反，重力被实现为类似于虚拟弹力的每个节点连接到布局尺寸(size)的中心的弱几何约束。这种方法具有很好的特性：靠近布局的中心，引力强度几乎为零，避免了布局的任何局部变形；当节点将被推远离中心，引力强度与距离成线性比例变强。因此，重力总是会某个阈值克服斥力电荷势力，以防止断开连接的节点逃逸出布局。 重力可以通过设置引力强度为0来禁止。如果禁用重力，建议你实现一些其他的几何约束，以防止逃逸布局，如在布局的范围内制约它们的节点。

force.**nodes**([*nodes*])

如果节点*nodes* 被指定，设置布局的相关节点为指定的nodes 数组。如果未指定节点*nodes* ，则返回当前数组，默认为空数组。每个节点具有以下属性：

- index - *nodes* 数组节点的索引（从零开始）。
- x - 当前节点的x坐标位置。
- y - 当前节点的位置y坐标。
- px - 前一个节点位置的x坐标。
- py - 前一个节点位置的y坐标。
- fixed - 一个布尔值，表示节点位置是否被锁定。
- weight - 节点权重；相关联的链接的数目。

这些属性不必在传递节点给布局之前进行设置；如果他们都没有设置，合适的默认值将在布局进行初始化start时调用，但是，要知道，如果你的节点上存储有其他数据，你的数据属性不应该与上面使用的布局属性冲突。

force.**links**([*links*])

如果指定了链接*links* ，设置布局的相关链接为指定的links 数组。如果没有指定链接*links* 时，返回当前数组，默认为空数组。每个链接都有以下属性：

- source - 源节点（节点中的元素）。
- target - 目标节点（节点中的元素）。

注意：在源和目标属性的值可初始化为*nodes* 数组的索引；这些将被替换为调用开始函数(start)之后的引用。链接对象可能有你指定的其他字段，这个数据可以用来计算链接强度strength和距离distance，基于每个连接的基础使用一个访问函数

force.**start**()

启动模拟；当首次创建布局时此方法必须被调用，然后分配节点和链接。此外，每当节点或链接发生变化它应当再次调用。在内部，布局使用冷却参数*alpha*控制布局的温度：当物理模拟收敛为稳定的布局，温度就下降，造成节点移动速度比较慢。最终，*alpha*下降到低于阈值，模拟完全停止，释放CPU资源，避免电池电量的消耗。布局可以使用恢复(resume)或重新启动重新加热；使用拖曳(drag)的行为时，会自动出现这种情况。

在开始时，布局初始化相关节点上的各种属性。每个节点的索引(*index*)是通过遍历数组，从零开始计算。初始的*x*和*y*坐标，如果尚未在外部设置为以有效的数字，通过检测相邻节点计算：如果链接的节点已经在*x*或*y*的初始位置时，相应的坐标被施加到新节点。这当新节点被添加时增加图形布局的稳定性，而不是使用默认值（在布局的尺寸之内随机初始化位置）。前一*px*和*py*位置设置为初始位置，如果尚未设置，给新节点一个零初始速度。最后，固定布尔默认为false。

布局还在相关链接上初始化源*source* 和目标*target* 属性：为方便起见，这些属性可以被指定为一个数字索引，而不是直接的链接，使得节点和链接可以从JSON文件或其他静态的描述中读取。如果这些属性是数字，导入链接的源和目标属性仅替换为nodes 中相应的实体；因此，现有链接中的这些属性当布局被重新启动时不受影响。链接距离distances和强度strengths也在开始时重新计算 .

force.**alpha**([*value*])

获取或设置力布局的冷却参数：*alpha*。如果值*value* 已指定，设置alpha为指定的值并返回力布局。如果值大于零，这个方法也将重新启动力布局（如果它尚未运行），分发一个"启动"事件启用节拍定时器。如果值为非正，且力布局正在运行，这个方法将在下一个tick停止力布局并分派"结束"事件。如果未指定值，则该方法返回当前alpha值。

force.**resume**()

相当于：

```
force.alpha(.1);
```

设置冷却参数alpha为0.1。此方法设置内部的alpha参数设置为0.1，然后重新启动定时器(timer)。 通常情况下，你不需要直接调用此方法；它是通过start自动调用。它也可以通过拖动动作drag自动调用。

force.**stop**()

相当于:

```
force.alpha(0);
```

终止模拟，冷却参数*alpha*设定为零。这可以用来显式地停止模拟，例如，如果你要展示的动画或允许其他的互动。如果你没有明确停止布局，它仍然会自动在布局的冷却参数衰变后低于某个阈值后停止。

force.**tick**()

执行力布局仿真一步。这种方法可以在配合使用start 和 stop 来计算静态布局。例如：

```
force.start();
for (var i = 0; i < n; ++i) force.tick();
force.stop();
```

迭代次数取决于图形的大小和复杂性。初始位置的选择也可以对如何快速将图形收敛于一个很好的解产生显着影响。例如，下面的节点沿对角线排列：

```
var n = nodes.length;
nodes.forEach(function(d, i) {
  d.x = d.y = width / n * i;
});
```

如果不手动初始化位置，力布局将它们随机初始化，导致有些不可预知的行为。

force.**on**(*type*, *listener*)

注册指定的监听器*listener* 为力布局指定类型的事件。目前，仅支持"start"，"tick"和"end"事件。"tick"事件将被指派为模拟的每个tick。监听节拍事件来更新节点和链接的显示位置。例如，如果你最初显示的节点和链接，象这样：

```
var link = vis.selectAll("line")
    .data(links)
  .enter().append("line");

var node = vis.selectAll("circle")
    .data(nodes)
  .enter().append("circle")
    .attr("r", 5);
```

您可以在tick设置他们的位置:

```
force.on("tick", function() {
  link.attr("x1", function(d) { return d.source.x; })
      .attr("y1", function(d) { return d.source.y; })
      .attr("x2", function(d) { return d.target.x; })
      .attr("y2", function(d) { return d.target.y; });

  node.attr("cx", function(d) { return d.x; })
      .attr("cy", function(d) { return d.y; });
});
```

在这个案例中，我们已经存储选择的 `node` 和 `link` 在初始化中，这样我们就不需要重新选择每个节点的tick。如果你愿意，你可以不同地显示节点和链接；例如，您可以使用符号(symbols)而不是圆形。

当模拟内部的alpha冷却低于阈值(0.005)时，"end"事件就被调度并被设为零。

force.**drag**()

绑定一个行为允许交互式拖动到节点，无论是使用鼠标或触摸。和节点中的call操作符一起使用；例如，初始化时调用 `node.call(force.drag)` 。拖动事件在鼠标滑过时设置节点的固定属性，这样，只要鼠标移动到某个节点，它停止不动。鼠标滑过而不是鼠标按下时固定，使得它更容易捕捉移动节点。当接收到一个鼠标按下事件在每个后续的鼠标移动直到鼠标弹起，当前鼠标位置设置为节点的中心。此外，每个鼠标移动触发一个力布局重新开始(resume)，再加热模拟。如果你想拖拖之后节点保持固定，在头动开始时设置*fixed* 属性为 `true` ，如粘力布局(sticky force layout)的例子。

实现注意：在鼠标移动和鼠标弹起事件监听器已注册当前窗口上，这样，当用户开始拖动节点，他们可以继续拖动节点，即使鼠标离开窗口。每个事件监听器使用"force"命名空间，以避免和其他你可能想绑定到节点或窗体上的事件监

听器冲突。如果节点被拖动事件移除了，随后的点击事件将被抓获到的最后一个鼠标抬起事件触发，你可以忽略这些点击和拖动通过查看是否默认的行为被阻止了。

```
selection.on("click", function(d) {
  if (d3.event.defaultPrevented) return; // ignore drag
  otherwiseDoAwesomeThing();
});
```

详见: collapsible force layout 和 divergent forces

咕噜译 2014-11-30 20:37:37