# Force Layout

[Wiki](https://github.com/mbostock/d3/wiki/Home) ▸ [API Reference](https://github.com/mbostock/d3/wiki/API-Reference) ▸ [Layouts](https://github.com/mbostock/d3/wiki/Layouts) ▸ **Force Layout**

一个灵活的受力导向图布局 实现 using 位置[Verlet integration](http://en.wikipedia.org/wiki/Verlet_integration) 允许 [simple constraints](http://www.csse.monash.edu.au/~tdwyer/Dwyer2009FastConstraints.pdf). 有关物理模拟的更多信息，请参见[Thomas Jakobsen](http://www.gamasutra.com/resource_guide/20030121/jacobson_pfv.htm). 这个实例是利用[quadtree](https://github.com/mbostock/d3/wiki/Quadtree-Geom)加速电荷相互作用使用[Barnes–Hut approximation](http://en.wikipedia.org/wiki/Barnes%E2%80%93Hut_simulation). 除了电荷[charge](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-charge) 力, pseudo-[gravity](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-gravity) force保持节点中心在可见区域并且避免expulsion of disconnected subgraphs, while links are fixed-[distance](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-distance) geometric constraints. 额外的自定义力和约束可应用于"tick" 事件, 只需通过更新节点的x和y属性。

http://bl.ocks.org/mbostock/4062045

例子:

* [divergent forces](http://bl.ocks.org/mbostock/1021841)
* [multiple foci](http://bl.ocks.org/mbostock/1021953)
* [graph constructor](http://bl.ocks.org/mbostock/929623)
* [force-directed tree](http://bl.ocks.org/mbostock/1062288)
* [force-directed symbols](http://bl.ocks.org/mbostock/1062383)
* [force-directed images and labels](http://bl.ocks.org/mbostock/950642)
* [force-directed states](http://bl.ocks.org/mbostock/1073373)
* [sticky force layout](http://bl.ocks.org/mbostock/3750558)

像D3的其他类, 布局s follow the method chaining pattern where setter methods return the 布局 itself, allowing multiple setters to be invoked in a concise statement.有别于其他的布局实现which are stateless,受力导向图布局 keeps a reference to the associated 节点 and links internally; 因此, 一个给定的受力导向图布局实例只可以用一个单一的数据集。.

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-force) d3.layout.**force**()

构造一个默认设置的受力导向图: size 1×1, link strength 1, friction 0.9, distance 20, charge strength -30, gravity strength 0.1, and theta parameter 0.8. 默认的节点和链接是空数组, 并且当布局开始时, the internal alpha cooling 参数被设置成0.1. 通用模式 for构建受力导向图布局 is 将所有配置属性, 然后调用开始[start](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start):

var force = d3.layout.force()

.nodes(nodes)

.links(links)

.size([w, h])

.start();

需要注意的是, 像D3的其他布局, 受力导向图布局不要求特定的可视化representation.最常见, 节点被映射到 SVG圆形元素, 并且链接被映射为SVG线元素，但你也可以显示节点作为[symbols](http://bl.ocks.org/mbostock/1062383) 或 [images](http://bl.ocks.org/mbostock/950642).

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-size) force.**size**([*size*])

如果大小被指定, 通过指定的两个数字元素的数组来代表 *x* 和 *y*设置可用的布局大小。如果大小未被指定, 返回当前大小, 默认为1×1. 大小影响受力导向图的两个方面: 受力中心和初识的随机位置. 重心只是 [ *x* / 2, *y* / 2 ]. 当节点被添加到该受力导向图布局, 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]的随机分布进行初始化。

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-linkDistance) force.**linkDistance**([*distance*])

如果 *距离* 被指定, 设定链接节点间的目标距离为指定的值. 如果 *距离* 未被指定，返回布局的当前链路距离，默认为20，如果*距离*是恒定的，那么所有的链接是相同的距离。否则，如果距离是一个函数，则该函数为每个链接（按顺序）求值，传递链接和它的索引，用 this作为受力导向图布局; 该函数的返回值被用来设置每个链接距离。当布局开始[starts](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start)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](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-size).

链接不作为"spring forces", 正如在一般受力导向图布局中，但是作为弱几何约束 。对于布局的每一个tick，每对连接的节点之间的距离与目标距离进行巨酸和比较; 链接然后朝向彼此移动或彼此远离,以收敛到所需的距离. 这种方法constraints relaxation on top of position Verlet integration is vastly more stable than previous methods using spring forces, 并且还允许灵活实施 of [other constraints](http://www.csse.monash.edu.au/~tdwyer/Dwyer2009FastConstraints.pdf)在Tick事件监听器, 如 hierarchical layering.

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-linkStrength) force.**linkStrength**([*strength*])

如果强度被指定, 设置链接间强度（刚性）为[0,1]范围内的指定的值. 如果强度未被指定,, 返回布局的当前链路强度，默认为1，如果强度是一个常数，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，传递链接和它的索引，具有这方面的动力布局;该函数的返回值被用来设置每个链接的强度。每当布局开始[starts](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start).

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-friction) force.**friction**([*friction*])

如果摩擦系数被指定, 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是误导性的;它不对应摩擦的标准物理系数[coefficient of friction](http://en.wikipedia.org/wiki/Friction#Coefficient_of_friction)。相反，它更接近速度衰减：在模拟的每个tick，粒子速度被指定的摩擦缩放。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。超出范围的值[0,1]不推荐，可影响能稳定。

[#](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-charge) force.**charge**([*charge*])

如果电荷强度被指定，设置电荷强度为指定的值。如果电荷强度为被指定，返回电流充电的实力，其默认值为-30。如果电荷强是恒定的，那么所有节点都具有相同的电荷。否则，如果电荷强是一个函数，则该函数每个节点（按顺序）求值，传递节点和它的索引，在此this情况下作为力布局;该函数的返回值被用于设置每个节点的电荷。每当布局[starts](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start)。.

负值导致节点排斥, while a positive value results in node attraction. 对于图形布局，应使用负值;对于 [*n*-body simulation](http://mbostock.github.com/protovis/ex/nbody.html)模拟,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。Charge forces are implemented efficiently via the [Barnes–Hut algorithm](http://arborjs.org/docs/barnes-hut), computing a [quadtree](https://github.com/mbostock/d3/wiki/Quadtree-Geom) for each tick. Setting the charge force to zero disables computation of the quadtree, which can noticeably improve performance，如果你不需要n-body forces。

[#](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-chargeDistance) force.**chargeDistance**([*distance*])

如果距离被指定，sets the maximum distance over which charge forces are applied. 如果距离未被指定，返回当前最大充电距离，默认为无穷大. 指定一个finite 电荷距离提高受力导向图的性能和produces a more localized layout; 限定距离的 charge forces are 尤其有用in conjunction with custom gravity. 有关示例, 请参阅 [“Constellations of Directors and their Stars”](http://www.nytimes.com/newsgraphics/2013/09/07/director-star-chart/) (*The New York Times*).

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-theta) force.**theta**([*theta*])

如果*theta*被指定, sets the Barnes–Hut approximation criterion为指定的值 如果theta未被指定, 返回当前值，默认为0.8。不同链接, 这不仅影响两个连接的节点， the charge force is 全局性的: 每个节点会影响所有其他节点，即使他们是在disconnected subgraphs.

为了避免 quadratic performance slowdown for large graphs,受力导向图布局使用 the [Barnes–Hut approximation](http://en.wikipedia.org/wiki/Barnes-Hut_simulation) which takes O(*n* log *n*) per tick. 对于每一个tick, a quadtree 被创建用于存储当前节点的位置; 那么对于每个节点，总charge force of 给定节点上的所有其它节点被计算. For clusters of过远的节点, the charge force is approximated by treating the distance cluster of nodes 作为一个单一的大的节点. *Theta* determines计算精度:如果面积的比例a象限in the quadtree to 距离节点和象限的中心之间of mass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。

[#](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-gravity) force.**gravity**([*gravity*])

如果指定了重力，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](http://en.wikipedia.org/wiki/Gravitation)（也可以用一个正电荷参数进行仿真[charge](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-charge) parameter) 。相反, 重力被实现为类似于虚拟弹簧的每个节点连接到布局的[size](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-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; as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, 重力将克服电荷斥力在某个阀值 preventing disconnected 节点超出布局.

重力可以通过引力强度设置为0来禁止. 如果禁用重力，所以建议你实现一些其他的几何约束，以防止逃逸的布局，如布局的范围内制约它们的节点。

[#](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-nodes) force.**nodes**([*nodes*])

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

* index -点数组中节点的从零开始的索引。
* 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;相关联的链路的数目。

这些属性不必通过节点的布局之前进行设置;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](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-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

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-links) force.**links**([*links*])

如果指定了链接，设置布局的相关链接到指定的数组。如果没有指定链接时，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最初可指定为索引到节点数组; 这些将被引用所替换，在初始化[start](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start)被调用时. 链接对象可能有你指定的其他领域，这个数据可以用来计算链接强度[strength](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-linkStrength) 和距离 [distance](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-linkDistance)基于每个链路的基础使用一个访问函数

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-start) force.**start**()

Starts the simulation; 首次创建布局时此方法必须被调用，然后后分配节点和链接. 此外，每当节点或链路发生变化它应当再次调用。. Internally, the layout uses a cooling parameter *alpha* which controls the layout temperature: as the physical simulation converges on a stable layout, the temperature drops, 造成节点移动速度比较慢。最终, *alpha* 下降到低于阈值和模拟完全停止, 释放CPU资源，避免电池电量的消耗.. 布局可以使用恢复或重新启动重新be reheated; 使用拖曳的行为时，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。每个节点的索引是通过遍历数组，从零开始计算。的初始x和y坐标，如果尚未设置外部，以有效的数字，计算通过检测相邻节点：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。这增加的图形布局的稳定性，当新节点被添加，而不是使用其内的布局的随机初始化位置的默认大小[size](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-size). 前一px和py位置设置为初始位置时，如果尚未设置，提供了新的节点零的初始速度。最后，固定布尔默认为false。

布局还在相关链接上初始化源和目标属性: 为方便起见, 这些属性可以被指定为一个数字索引，而不是直接的联系,使得节点和链接可以从JSON文件或其他静态的描述中读取， 源和目标属性on 导入链接仅替换为相应的节点项 ，如果这些属性是数字; 因此, 这些属性在现有链接不受影响 当布局被重新启动. 链路距离[distances](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-linkDistance)和强度[strengths](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-linkStrength)也在开始时重新计算 .

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-alpha) force.**alpha**([*value*])

获取或设置力布局的cooling参数, *alpha*. 如果值已指定，设置*alpha*为指定的值并返回力布局。如果值大于零，这种方法也将重新启动力布局，如果它尚未运行dispatching一个“启动”事件and enabling the tick timer. 如果值为非正，且力布局正在运行, 这个方法将停止力布局在下一个tick和 dispatches“结束”事件。。. 如果未指定值，则该方法返回当前alpha值

[#](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-resume) force.**resume**()

相当于:

force.alpha(.1);

设置cooling参数*alpha* to 0.1. 此方法设置内部的alpha参数设置为0.1，然后重新启动[timer](https://github.com/mbostock/d3/wiki/Transitions#wiki-d3_timer). 通常情况下，你不需要直接调用此方法;它是由自动初始化[start](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start)调用. 它也可以通过拖动动作[drag](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-drag)自动调用。.

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-stop) force.**stop**()

相当于:

force.alpha(0);

终止simulation，cooling参数的*alpha*设定为零。这可以用来显式地停止simulation，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，它仍然会自动布局的cooling参数衰变后低于某个阈值停止,

[#](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-tick) force.**tick**()

执行力布局仿真一步。这种方法可以在配合使用[start](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-start) and [stop](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-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的图形收敛于一个很好的解决方案产生重大影响。例如，下面的节点沿对角线:

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。.

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-on) force.**on**(*type*, *listener*)

注册指定侦听器，以从力布局接收指定类型的事件. 目前，仅"start", "tick", and "end"事件的支持。. "tick"事件将被指派为模拟的每个tick. Listen to tick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; });

});

在这种情况下，我们已经存储的选择的 node and link on初始化, 这样我们就不需要重新选择每个节点的tick。如果你愿意，你可以显示节点和链接不同;例如，您可以使用符号而不是圆形。.

当the simulations internal的alpha冷却参数达到零The "end"事件被调度.

[#](https://github.com/mbostock/d3/wiki/Force-Layout" \l "wiki-drag) 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. Use this in 一起 with the [call](https://github.com/mbostock/d3/wiki/Selections#wiki-call) operator on the nodes; 例如, say node.call(force.drag) 在初始化. 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. 固定on 鼠标滑过 而不是鼠标按下, 使得它更容易捕捉移动节点. 当接收到一个鼠标按下事件, and on each subsequent **鼠标移动**  until **鼠标弹起**, 当前鼠标位置设置为节点的中心. 此外, each **鼠标移动**  triggers a [resume](https://github.com/mbostock/d3/wiki/Force-Layout#wiki-resume) of the force layout, reheating the simulation. If you want dragged nodes to remain fixed after dragging, set the *fixed* attribute to true on *dragstart*, as in the [sticky force layout](http://bl.ocks.org/mbostock/3750558) example.

实现注意事项：在鼠标移动 和鼠标弹起事件侦听器已注册当前窗口上，这样，当用户开始拖动节点，他们可以继续拖东节点即使鼠标离开窗口. 每个使用的"force"的命名空间,事件侦听器，从而避免 avoid collision with其它事件侦听你想绑定到节点或到窗口。如果一个节点被拖拽的行为移动 t随后的click事件that would 触发通过最终的鼠标弹起 被捕获并默认行为阻止. 如果您注册一个click事件侦听器, 你可以忽略这些点击拖动 by seeing 如果默认的行为被阻止:

selection.on("click", function(d) {

if (d3.event.defaultPrevented) return; // ignore drag

otherwiseDoAwesomeThing();

});

See the [collapsible force layout](http://bl.ocks.org/mbostock/1093130) and [divergent forces](http://bl.ocks.org/mbostock/1021841) for examples.