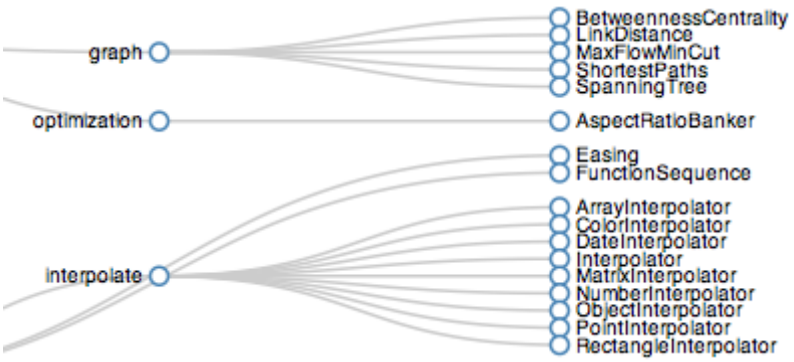
> *Wiki* ‣ *[[API--中文手册]]* ‣ *[[布局]]* ‣ ***簇布局***

- 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

簇布局(cluster layout)可以产生树状图(dendrograms)：将树的叶节点放在同一深度的节点连接图。例如，簇布局可以用来在软件包层次结构中组织类：



同D3中的其他类一样，布局遵循方法链模式，在该模式下设置器(setter)方法返回布局本身，允许使用简单语句调用多个setter。

d3.layout.**cluster**()

使用默认设置创建新的簇布局：默认排序为空；默认子访问器假定每个输入数据为带子数组的对象；默认分离函数在同层级使用一个节点宽度，在不同层级使用两个节点宽度；默认尺寸为1×1。

cluster(*root*)
cluster.**nodes**(*root*)

运行簇布局，返回节点数组及指定的根节点。簇数组为D3分层布局家族的一部分。这些布局具有相同的基本结构：布局的输入参数为分层的根节点(*root*)，输出返回值为一个数组，表示计算过的所有节点的位置。每个节点都有各自属性：

- 上层 - 父节点，在根节点时为空。
- 下层 - 子节点数组，在叶节点时为空。
- 深度 - 节点深度，从根节点计算，值从0开始。
- x - 节点位置的*x*坐标值。
- y - 节点位置的*y*坐标。

虽然布局在*x*和*y*轴有值存在，但这表示一个任意坐标系；例如，可以将*x*坐标视为直径，将*y*坐标当做角，从而形成一个射线，而非笛卡尔坐标系布局。

cluster.**links**(*nodes*)

指定节点数组，如以节点形式(nodes)返回的数组，返回对象节点表示每个节点中父节点同子节点之间的关系。叶节点没有任何关系。每个节点都是一个具有两个属性的对象：

- source -父节点（如上述所示）。
- target -子节点。

该方法在获取一组关系描述时很有效果，通常与对角(diagonal)图形生成器共同使用。例如：

```
svg.selectAll("path")
    .data(cluster.links(nodes))
  .enter().append("path")
    .attr("d", d3.svg.diagonal());
```

cluster.**children**([*children*])

如果子节点*children*已经指定，则设定子节点访问器函数。如未指定*children*，则返回当前子节点访问器函数，该函数将输入数据默认为带子数组的对象：

```
function children(d) {
  return d.children;
}
```

通常，使用d3.json可以方便地加载节点分层，并将输入分层表示为一个嵌套JSON对象。例如：

```
{
 "name": "flare",
 "children": [
  {
   "name": "analytics",
   "children": [
    {
     "name": "cluster",
     "children": [
      {"name": "AgglomerativeCluster", "size": 3938},
      {"name": "CommunityStructure", "size": 3812},
      {"name": "MergeEdge", "size": 743}
     ]
    },
    {
     "name": "graph",
     "children": [
      {"name": "BetweennessCentrality", "size": 3534},
      {"name": "LinkDistance", "size": 5731}
     ]
    }
   ]
  }
 ]
}
```

在分层中，子访问器在根节点首先被调用。如果访问器返回值为空，则该节点在布局遍历结束时被假定为叶节点。否则，访问器需要返回数据元素数组，来表示子节点。

cluster.**sort**([*comparator*])

如已指定*comparator*，则使用指定的comparator函数设置布局中同级节点排序顺序。如*comparator*未指定，则返回当前分组的排序顺序，默认值为空。为每对节点，调用comparator函数。 comparator的默认值为空，此时采用三种遍历顺序，排序被禁用。例如，按照输入数据的字符串名对同层节点以降序顺序排序，即：

```
function comparator(a, b) {
  return d3.ascending(a.name, b.name);
}
```

See d3.ascending or d3.descending for details. 详见 d3.ascending 或d3.descending.

cluster.**separation**([*separation*])

如果已经指定*separation*，使用指定的函数计算相邻节点的间距。如果未指定*separation*，则返回当前间距函数。默认情况下，该函数为：

```
function separation(a, b) {
  return a.parent == b.parent ? 1 : 2;
}
```

存在一个更加适合于射线布局的变动，可以根据直径大小相应减少间距：

```
function separation(a, b) {
  return (a.parent == b.parent ? 1 : 2) / a.depth;
}
```

两个相邻节点*a*和*b*传递到separation函数，且必须返回节点间期望的间距。节点通常是同级的，虽然，这些节点也可能属于相近关系（或更远的关系），如果布局将这些节点临近放置的话。

cluster.**size**([*size*])

如果已经指定*size*，则将可用布局尺寸设定为指定的二元数组，以*x*和*y*来表示。如果尺寸没有指定，则返回当前尺寸，默认值为1×1，如果nodeSize正在使用中，则默认值为空。虽然布局在*x*和y轴都有数值，但该坐标系可以是任意坐标系。例如，创建一个射线布局，其中树的广度（*x*）用角度来测量, 树的深度（*y*）表示半径*r*, 单位为像素，即 [360, *r*]。

cluster.**nodeSize**([*nodeSize*])

如果*nodeSize*已经指定，以二元数组*x*和*y*的形式返回每个节点的固定尺寸。如果*nodeSize*没有指定，则返回当前节点尺寸，默认值为空，表示布局尺寸总体固定，可以使用size来获得。

cluster.**value**([*value*])

如果已经指定*value*，则用指定的函数设定值访问器。如果尚未指定*value*，则返回当前值访问器，默认值为空，表示值属性没有计算。指定之后，每次输入数据元，都会调用值访问器，并且必须返回一个用以表示节点数值的数字。该值对集群布局没有影响，但它是分层布局所提供的通用功能。

- 张烁译 20140430
- 咕噜校对 2014-11-30 10:42:08