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

- 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

分层布局是一种*抽象*布局，不直接使用。但它允许在不同的分层布局中共享节点。请看下列例子：

- Cluster - 簇实体构成树状图。
- Pack - 使用递归圆填充法创建一个分层布局。
- Partition - 将节点树递归划分成辐射状或环状。
- Tree - 将节点树整齐放置。
- Treemap - 使用递归空间划分显示节点树。

虽然不是分层布局，但捆布局可以同分层共同使用。

d3.layout.**hierarchy**()

使用以下默认设置创建新的分层布局：默认排序顺序为值的降序排列；默认值访问器假定每个输入数据为一个含数值属性的对象；默认子访问器假定每个输入数据为一个含子数列的对象。

hierarchy(*root*)

运行分层布局，返回节点数组及指定的根节点。布局的输入参数为分层的根节点(*root* node)，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：

- parent -父节点，在根节点时为空。
- children -子节点数组，在叶节点时为空。
- value -节点值，由值访问器返回。
- depth -节点深度，从根节点计算，值从0开始。

此外，多数分层布局也会计算节点的*x*和*y*的位置；详见实现类。

hierarchy.**links**(*nodes*)

指定节点数组*nodes*返回一个对象数组，该数组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：

- 源-父节点（如上述所示）。
- 目标-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(partition.links(nodes))
  .enter().append("path")
    .attr("d", d3.svg.diagonal());
```

hierarchy.**children**([*accessor*])

如果*accessor*已经指定，则设定子访问器函数。如未指定，则返回当前子访问器函数，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源数组，来表示子节点。参数*node*和*depth*都需要调用访问器。

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

如已指定*comparator*，则使用指定的comparator函数设定布局的同级节点节点顺序。如*comparator*未指定，则返回当前分组的排序顺序，默认值为按照输入数据的字符串名对节点的降序顺序排序：

```
function comparator(a, b) {
  return b.value - a.value;
}
```

为每对节点，调用comparator函数。零comparator禁用排序，使用树遍历顺序。comparator函数也可以通过d3.ascending或d3.descending实现。

hierarchy.**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。默认访问器假定输入数据为一个具有数值属性的对象：

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

每次输入数据元素，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。对于区域布局，如树图，该值用于设定每个节点值相应的面积。对于其他布局，该值对簇布局没有影响。

hierarchy.**revalue**(*root*)

对于一棵指定的树，从根*root*开始重计算每个节点的值，但不需对子节点进行重新排序或重新计算。该方法可以用于对每个节点值进行重新计算，但又不必对分层做出任何结构改变。最初，该方法是用来支持sticky treemaps的。

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