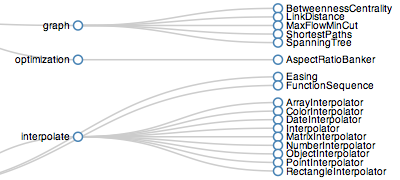
**Cluster Layout**

[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) ▸ [Hierarchy](https://github.com/mbostock/d3/wiki/Hierarchy-Layout) ▸ **Cluster Layout**

The **cluster layout** produces [dendrograms](http://en.wikipedia.org/wiki/Dendrogram): node-link diagrams that place leaf nodes of the tree at the same depth. For example, a cluster layout can be used to organize software classes in a package hierarchy:

聚类布局可以产生聚类图：将树的叶节点放在同一深度的节点连接图。例如，聚类布局可以用来在包层次结构中组织软件类：



Like other classes in D3, layouts follow the method chaining pattern where setter methods return the layout itself, allowing multiple setters to be invoked in a concise statement.

同D3中的其他类一样，布局遵循方法连接模式，在该模式下setter方法返回布局本身，允许使用简单语句调用多个setter。

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-cluster) d3.layout.cluster()

Creates a new cluster layout with the default settings: the default sort order is null; the default children accessor assumes each input data is an object with a children array; the default separation function uses one node width for siblings, and two node widths for non-siblings; the default size is 1×1.

使用默认设置创建新的聚类布局：默认分类次序为空；默认子存取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；默认大小为1×1。

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-_cluster) cluster(root) [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-nodes) cluster.nodes(root)

Runs the cluster layout, returning the array of nodes associated with the specified root node. The cluster layout is part of D3's family of [hierarchical](https://github.com/mbostock/d3/wiki/Hierarchy-Layout) layouts. These layouts follow the same basic structure: the input argument to the layout is the root node of the hierarchy, and the output return value is an array representing the computed positions of all nodes. Several attributes are populated on each node:

运行聚类布局，返回节点数组及指定的根节点。聚类数组为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，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 - the parent node, or null for the root.
* children - the array of child nodes, or null for leaf nodes.
* depth - the depth of the node, starting at 0 for the root.
* x - the computed x-coordinate of the node position.
* y - the computed y-coordinate of the node position.
* 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。

Although the layout has a size in x and y, this represents an arbitrary coordinate system; for example, you can treat x as a radius and y as an angle to produce a radial rather than Cartesian layout.

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。

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-links) cluster.links(nodes)

Given the specified array of nodes, such as those returned by [nodes](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-nodes), returns an array of objects representing the links from parent to child for each node. Leaf nodes will not have any links. Each link is an object with two attributes:

* source - the parent node (as described above).
* target - the child node.

This method is useful for retrieving a set of link descriptions suitable for display, often in conjunction with the [diagonal](https://github.com/mbostock/d3/wiki/SVG-Shapes#wiki-diagonal) shape generator. For example:

指定节点数组，如以节点形式返回的数组，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：

* 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，通常与对焦图形发生器共同使用。例如：

svg.selectAll("path")

.data(cluster.links(nodes))

.enter().append("path")

.attr("d", d3.svg.diagonal());

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-children) cluster.children([children])

If children is specified, sets the specified children accessor function. If children is not specified, returns the current children accessor function, which by default assumes that the input data is an object with a children array:

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

function children(d) {

return d.children;

}

Often, it is convenient to load the node hierarchy using [d3.json](https://github.com/mbostock/d3/wiki/Requests#wiki-d3_json), and represent the input hierarchy as a nested [JSON](http://json.org) object. For example:

通常，使用[d3.json](https://github.com/mbostock/d3/wiki/Requests#wiki-d3_json)可以方便地加载节点分级，并将输入分级表示为一个嵌套[JSON](http://json.org)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}

]

}

]

}

]

}

The children accessor is first invoked for root node in the hierarchy. If the accessor returns null, then the node is assumed to be a leaf node at the layout traversal terminates. Otherwise, the accessor should return an array of data elements representing the child nodes.

在分级中，子存取器在根节点首先被调用。如果存取器返回值为空，则该节点在布局遍历结束时被假定为叶节点。否则，存取器需要返回数据源数组，来表示子节点。

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-sort) cluster.sort([comparator])

If comparator is specified, sets the sort order of sibling nodes for the layout using the specified comparator function. If comparator is not specified, returns the current group sort order, which defaults to null for no sorting. The comparator function is invoked for pairs of nodes, being passed the input data for each node. The default comparator is null, which disables sorting and uses tree traversal order. For example, to sort sibling nodes in descending order by the associated input data's string name attribute, say:

如已指定comparator，则使用指定的comparator函数返回同级节点顺序。如comparator未指定，则返回当前分组的分类顺序，默认值为空。当节点成对出现时，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](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_ascending) or [d3.descending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_descending) for details.

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-separation) cluster.separation([separation])

If separation is specified, uses the specified function to compute separation between neighboring nodes. If separation is not specified, returns the current separation function, which defaults to:

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;

}

A variation that is more appropriate for radial layouts reduces the separation gap proportionally to the radius:

存在一个更加适合于射线布局的变分，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;

}

The separation function is passed two neighboring nodes a and b, and must return the desired separation between nodes. The nodes are typically siblings, though the nodes may also be cousins (or even more distant relations) if the layout decides to place such nodes adjacent.

两个相邻节点a和b传递到separation函数，且必须返回节点间的期望间距。节点属于同一级别，但如果布局将这些节点临近放置的话，也可能属于相近关系（或更远的关系）。

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

If size is specified, sets the available layout size to the specified two-element array of numbers representing x and y. If size is not specified, returns the current size, which defaults to 1×1, or null if a [nodeSize](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-nodeSize) is in use. Although the layout has a size in x and y, this represents an arbitrary coordinate system. For example, to produce a radial layout where the tree breadth (x) in measured in degrees, and the tree depth (y) is a radius r in pixels, say [360, r].

如果已经指定*size*，则将可用布局大小设定为指定的二元数组，以x和y来表示。如果大小没有指定，则返回当前大小，默认值为1×1， 如果[nodeSize](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-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]。

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-nodeSize) cluster.nodeSize([nodeSize])

If nodeSize is specified, sets a fixed size for each node as a two-element array of numbers representing x and y. If nodeSize is not specified, returns the current node size, which defaults to null, meaning that the layout has an overall fixed size, which can be retrieved using [size](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-size).

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大小。如果nodeSize没有指定，则返回当前节点大小，默认值为空，表示布局大小总体固定，可以使用[size](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-size)来获得。

#### [#](https://github.com/mbostock/d3/wiki/Cluster-Layout" \l "wiki-value) cluster.value([value])

If value is specified, sets the value accessor to the specified function. If value is not specified, returns the current value accessor which defaults to null, meaning that the value attribute is not computed. If specified, the value accessor is invoked for each input data element, and must return a number representing the numeric value of the node. This value has no effect on the cluster layout, but is generic functionality provided by hierarchy layouts.

如果已经指定*value，*则用指定的函数设定值存取器。如果尚未指定*value*，则返回当前值存取器，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，都会调用值存取器，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，但它是分级布局所提供的通用功能。

### [Hierarchy](https://github.com/mbostock/d3/wiki/Hierarchy-Layout)

* [d3.layout.hierarchy](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-hierarchy) - derive a custom hierarchical layout implementation.
* [hierarchy](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-_hierarchy) - alias for hierarchy.nodes.
* [hierarchy.nodes](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-nodes) - compute the layout and return the array of nodes.
* [hierarchy.links](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-links) - compute the parent-child links between tree nodes.
* [hierarchy.children](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-children) - get or set the accessor function for child nodes.
* [hierarchy.sort](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-sort) - get or set the comparator function for sibling nodes.
* [hierarchy.value](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-value) - get or set the value accessor function.
* [hierarchy.revalue](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-revalue) - recompute the hierarchy values.
* [d3.layout.hierarchy](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-hierarchy) - 取自自定义分级布局实现。
* [hierarchy](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-_hierarchy) - 分级节点的别名。
* [hierarchy.nodes](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-nodes) - 计算布局，返回节点数组。
* [hierarchy.links](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-links) -计算树节点之间的父-子关系。
* [hierarchy.children](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-children) -为子节点获取或设定存取器函数。
* [hierarchy.sort](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-sort) -为同层级节点获取或设定comparator函数。
* [hierarchy.value](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-value) -获取或设定值存取器函数。
* [hierarchy.revalue](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-revalue) -重新计算分级值。

**Hierarchy 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) ▸ **Hierarchy Layout**

The hierarchy layout is an abstract layout that is not used directly, but instead allows code sharing between multiple hierarchical layouts. For an implementation, see one of:

该分级布局是一种抽象布局，不直接使用。但它允许在不同的分级布局中共享节点。请看下列例子：

* [Cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout) - cluster entities into a dendrogram.
* [Pack](https://github.com/mbostock/d3/wiki/Pack-Layout) - produce a hierarchical layout using recursive circle-packing.
* [Partition](https://github.com/mbostock/d3/wiki/Partition-Layout) - recursively partition a node tree into a sunburst or icicle.
* [Tree](https://github.com/mbostock/d3/wiki/Tree-Layout) - position a tree of nodes tidily.
* [Treemap](https://github.com/mbostock/d3/wiki/Treemap-Layout) - use recursive spatial subdivision to display a tree of nodes.

Although not a hierarchy layout, the [bundle layout](https://github.com/mbostock/d3/wiki/Bundle-Layout) is also designed to work in conjunction with hierarchies.

* [Cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout) -聚类实体构成聚类图。
* [Pack](https://github.com/mbostock/d3/wiki/Pack-Layout) -使用递归圆填充法创建一个分级布局。
* [Partition](https://github.com/mbostock/d3/wiki/Partition-Layout) -将节点树递归划分成辐射状或环状。
* [Tree](https://github.com/mbostock/d3/wiki/Tree-Layout) -将节点树整齐放置。
* [Treemap](https://github.com/mbostock/d3/wiki/Treemap-Layout) -使用递归空间划分显示节点树。

虽然不是分级布局，但[bundle layout](https://github.com/mbostock/d3/wiki/Bundle-Layout)可以同分级共同作用。

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-hierarchy) d3.layout.hierarchy()

Creates a new hierarchy layout with the default settings: the default sort order is by descending value; the default value accessor assumes each input data is an object with a numeric value attribute; and the default children accessor assumes each input data is an object with a children array.

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

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout" \l "wiki-_hierarchy) hierarchy(root)

Runs the hierarchy layout, returning the array of nodes associated with the specified root node. The input argument to the layout is the root node of the hierarchy, and the output return value is an array representing the computed positions of all nodes. Several attributes are populated on each node:

* parent - the parent node, or null for the root.
* children - the array of child nodes, or null for leaf nodes.
* value - the node value, as returned by the value accessor.
* depth - the depth of the node, starting at 0 for the root.

In addition, most hierarchy layouts also compute x and y positions for nodes; see the implementing class for details.

运行分级布局，返回节点数组及指定的根节点。布局的输入参数为分级的根节点，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。
* 值-节点值， 由值存取器返回。
* 深度-节点深度，从根节点计算，值从0开始。

此外，多数分级布局也会计算节点的x和y的位置；详见实现类。

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-links) hierarchy.links(nodes)

Given the specified array of nodes returns an array of objects representing the links from parent to child for each node. Leaf nodes will not have any links. Each link is an object with two attributes:

* source - the parent node (as described above).
* target - the child node.

This method is useful for retrieving a set of link descriptions suitable for display, often in conjunction with the [diagonal](https://github.com/mbostock/d3/wiki/SVG-Shapes#wiki-diagonal) shape generator. For example:

指定节点数组返回一个对象数组，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，通常与对焦图形发生器共同使用。例如：

svg.selectAll("path")

.data(partition.links(nodes))

.enter().append("path")

.attr("d", d3.svg.diagonal());

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-children) hierarchy.children([accessor])

If accessor is specified, sets the specified children accessor function. If accessor is not specified, returns the current children accessor function, which by default assumes that the input data is an object with a children array:

如果*accessor*已经指定，则设定子存取器函数。如未指定，则返回当前子存取器函数，该函数将输入数据默认为带子数组的对象：

function children(d) {

return d.children;

}

Often, it is convenient to load the node hierarchy using [d3.json](https://github.com/mbostock/d3/wiki/Requests#wiki-d3_json), and represent the input hierarchy as a nested [JSON](http://json.org) object. For example:

通常，使用[d3.json](https://github.com/mbostock/d3/wiki/Requests#wiki-d3_json)可以方便地加载节点分级，并将输入分级表示为一个嵌套[JSON](http://json.org)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}

]

}

]

}

]

}

The children accessor is first invoked for root node in the hierarchy. If the accessor returns null, then the node is assumed to be a leaf node and the layout traversal terminates. Otherwise, the accessor should return an array of data elements representing the child nodes. The accessor is called with arguments node and depth.

在分级中，子存取器在根节点首先被调用。如果存取器返回值为空，则该节点在布局遍历结束时被假定为叶节点。否则，存取器需要返回数据源数组，来表示子节点。参数node 和 depth都需要调用存取器。

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-sort) hierarchy.sort([comparator])

If comparator is specified, sets the sort order of sibling nodes for the layout using the specified comparator function. If comparator is not specified, returns the current group sort order, which defaults to descending order by the associated input data's numeric value attribute:

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，则返回当前分组的分类顺序，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;

}

The comparator function is invoked for pairs of nodes, being passed the input data for each node. A null comparator disables sorting and uses tree traversal order. Comparator functions may also be implemented using [d3.ascending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_ascending) or [d3.descending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_descending).

当节点成对出现时，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](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_ascending) 或 [d3.descending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_descending)实现。

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout#wiki-value) hierarchy.value([value])

If value is specified, sets the value accessor to the specified function. If value is not specified, returns the current value accessor. The default accessor assumes that the input data is an object with a numeric value attribute:

如果已经指定*value，*则用指定的函数设定值存取器。如果尚未指定*value*，则返回当前值存取器。默认存取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;

}

The value accessor is invoked for each input data element, and must return a number representing the numeric value of the node. For area-proportional layouts such as treemaps, this value is used to set the area of each node proportionally to the value; for other hierarchical layouts, the value has no effect on the layout.

每次输入数据元，都会调用值存取器，并且必须返回一个用以表示节点数值的数字。对于面积比例布局，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，该值对集群布局没有影响。

#### [#](https://github.com/mbostock/d3/wiki/Hierarchy-Layout" \l "wiki-revalue) hierarchy.revalue(root)

Re-evaluates the values of each node in the specified tree starting at root, without re-sorting or recomputing the child nodes. This method can be used to recompute the values of each node without making any structural changes to the hierarchy. Primarily, it exists to support [sticky treemaps](https://github.com/mbostock/d3/wiki/Treemap-Layout#wiki-sticky).

对于一棵指定的树，从根开始重估每个节点的值，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，但又不必对分级做出任何结构改变。最初，该方法是用来支持[sticky treemaps](https://github.com/mbostock/d3/wiki/Treemap-Layout#wiki-sticky)的。

### [Histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout)

* [d3.layout.histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-histogram) - construct a new default histogram layout.
* [histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram) - compute the distribution of data using quantized bins.
* [histogram.value](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-value) - get or set the value accessor function.
* [histogram.range](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-range) - get or set the considered value range.
* [histogram.bins](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-bins) - specify how values are organized into bins.
* [histogram.frequency](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-frequency) - compute the distribution as counts or probabilities.
* 直方图
* [d3.layout.histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-histogram) -构建新的直方图布局。
* [histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram) -使用量化箱计算数据分布。
* [histogram.value](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-value) -获取或设置值存取器函数。
* [histogram.range](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-range) -获取或设置取值范围。
* [histogram.bins](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-bins) - 说明值组织成容器的方式。
* [histogram.frequency](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-frequency) -将分布计算成计数或概率。

**Histogram 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) ▸ **Histogram Layout**

A **histogram layout** shows the distribution of data by grouping discrete data points into bins. See [bl.ock 3048450](http://bl.ocks.org/mbostock/3048450) for example usage.

通过将离散数据点分组归纳到容器里，直方图布局可以用来表示数据分布。使用实例详见[bl.ock 3048450](http://bl.ocks.org/mbostock/3048450)。

#### [#](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-histogram) d3.layout.histogram()

Constructs a new histogram function with the default value accessor, range function, and bin function. By default, the histogram function returns frequencies. The returned layout object is both an object and a function. That is: you can call the layout like any other function, and the layout has additional methods that change its behavior. Like other classes in D3, layouts follow the method chaining pattern where setter methods return the layout itself, allowing multiple setters to be invoked in a concise statement.

使用默认值存取器、范围函数和容器函数， 构建新的直方图函数。默认条件下，直方图函数返回值为频率。返回布局对象既是一个对象，也是一个函数。即: 可以像调用其他函数一样调用该布局，并且布局用于其他手段改变自身行为。和D3中的其他类一样，布局遵循方法连接模式，在该模式下setter方法返回布局本身，允许使用简单语句调用多个setter。

#### [#](https://github.com/mbostock/d3/wiki/Histogram-Layout" \l "wiki-_histogram) histogram(values[, index])

Evaluates the histogram function on the specified array of values. An optional index may be specified, which is passed along to the range and bin function. The return value is an array of arrays: each element in the outer array represents a bin, and each bin contains the associated elements from the input values. In addition, each bin has three attributes:

在指定的values数组上孤独按直方图布局。 可以指定一个可选index, 传递给范围函数和容器函数。返回值为数组的数组：外部数组的每个元表示一个容器，每个容器包含输入values的相关元。此外，每个容器有三个属性：

* x - the lower bound of the bin (inclusive).
* dx - the width of the bin; x + dx is the upper bound (exclusive).
* y - the count (if [frequency](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-frequency) is true), or the probability (if frequency is false).

Note that the y attribute is the same as the length attribute, in frequency mode.

* x -箱的下界 （包含）。
* dx - 箱的宽度； x + dx为上届（不包含）。
* y - the count (if [frequency](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-frequency) is true), or the probability (if frequency is false). 计数（如果[frequency](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-frequency)为真），或概率（如果[frequency](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-frequency)为假）。

请注意，在频率方式上，y属性和长度属性相同。

#### [#](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-value) histogram.value([accessor])

Specifies how to extract a value from the associated data; accessor is a function which is invoked on each input value passed to [histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram), equivalent to calling values.map(accessor) before computing the histogram. The default value function is the built-in [Number](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Number), which is similar to the identity function. If accessor is not specified, returns the current value accessor.

详细说明从关联数据中提取值得方法； accessor是一个函数，每当输入值传递到[histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram) 时， 都需要调用该函数， 即等于在计算直方图之前调用values.map(accessor)。默认值函数为内置[Number](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Number)，同恒等函数类似。如果未指定accessor，则返回当前值存取器。

#### [#](https://github.com/mbostock/d3/wiki/Histogram-Layout" \l "wiki-range) histogram.range([range])

Specifies the range of the histogram. Values outside the specified range will be ignored. The range may be specified either as a two-element array representing the minimum and maximum value of the range, or as a function that returns the range given the array of values and the current index passed to [histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram). The default range is the extent ([minimum](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_min) and [maximum](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_max)) of the values. If range is not specified, returns the current range function.

指定直方图范围。忽略在指定范围之外的值。可以通过二元数组指定range，数组表示范围的最大值和最小值，或者将range指定为一个函数，该函数返回values数组和传递到[histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram)的当前索引。默认范围为值的长度（[minimum](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_min) 和 [maximum](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_max)）。如果未指定range，则返回当前范围函数。

[#](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-bins) histogram.**bins**()   
[#](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-bins) histogram.**bins**(*count*)   
[#](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-bins) histogram.**bins**(*thresholds*)   
[#](https://github.com/mbostock/d3/wiki/Histogram-Layout" \l "wiki-bins) histogram.**bins**(*function*)

Specifies how to bin values in the histogram. If no argument is specified, the current binning function is returned, which defaults to an implementation of [Sturges' formula](http://en.wikipedia.org/wiki/Histogram) that divides values into bins using uniformly-spaced values. If a count is specified, the value [range](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-range) is divided evenly into the specified number of bins.

If an array of thresholds is specified, it defines the value thresholds used to bin, starting with the leftmost (lowest) value and ending with rightmost (highest) value. The n + 1 thresholds specify n bins. Any values less than *thresholds[1]* will be placed in the first bin; likewise any values greater than or equal to *thresholds[thresholds.length - 2]* will be placed in the last bin. Thus, although the first and last threshold are not used to assign values to bins, they are still necessary to define the x property of the first bin and the dx property of the last bin, respectively.

Lastly, if a binning function is specified, it is invoked when the layout is passed data, being passed the current [range](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-range), the array of values and the current index passed to [histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram). This function must then return an array of thresholds as described in the previous paragraph.

详细说明如何将值归类到直方图中。如果没有指定参数，则返回当前bining函数，默认值为[Sturges' formula](http://en.wikipedia.org/wiki/Histogram)的一个实现，[Sturges' formula](http://en.wikipedia.org/wiki/Histogram)使用等间隔的值将值划分到不同的容器当中。如果已经指定count值，则将[range](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-range)的值均匀分布到指定数量的容器中。

如果已指定thresholds数组，则它定义了容器的极限值，从最左边的值（最小值）开始到最右边的值（最大值）。n + 1 thresholds指定了n个容器。任何小于*thresholds[1]*的值都将被放在第一个容器中；同理，任何大于或等于*thresholds[thresholds.length - 2]*的值将被放在最后一个容器中。因此，虽然第一个和最后一个极值并未分配到容器中，但他们对于定义第一个容器的x属性和最后一个容器的dx属性还是有必要存在的。

最后，如果已经指定容器function，该函数会在布局传递数据时调用，传递当前[range](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-range)，值得数列和当前索引传递到[histogram](https://github.com/mbostock/d3/wiki/Histogram-Layout#wiki-_histogram)。 该函数必须返回上文所述的thresholds数列。

#### [#](https://github.com/mbostock/d3/wiki/Histogram-Layout" \l "wiki-frequency) histogram.frequency([frequency])

Specifies whether the histogram's y value is a count (frequency) or a probability (density); the default is frequency. If frequency is not specified, returns the current frequency boolean.

指定直方图的y值是否是一个计数（频率）或概率（密度）； 默认值为频率。如果没有指定frequency，则返回当前频率的布尔值。