# ForceDirectedLayout(力导向布局)

**更新至version 2.0.4**

继承关系：

* **Layout**
  + **ForceDirectedLayout**

力导向布局对待视图，就好比视图中有一个物理对象系统，其中有包括电斥力、引力以及弹簧力，作用在这些物理对象上。

电斥力来自于顶点和其相邻顶点，并与顶点之间的距离成二次方程关系。引力来自于顶点本身，是恒定不变的。弹簧力仅来自于连接顶点的边线。

作用在一个顶点上的电斥力，是周围所有顶点的电荷量(**electricalCharge**)乘以电场(**electricalFieldX**, **electricalFieldY**)的总和，以及所有它附近顶点的电斥力除以距离平方。你可以为所有顶点设置默认电荷量(**defaultElectricalCharge**)。默认情况下没有电场，所以所有的力都是由相邻的带电顶点造成的。为了提高效率，最大影响距离(**infinityDistance**)确定了顶点之间的力能够影响的最大距离。

作用在一个顶点上的引力，是周围所有顶点的引力质量(**gravitationalMass**)乘以引力场(**gravitationalFieldX, gravitationalFieldY**)的总和。你可以为所有顶点设置默认引力质量(**defaultGravitationalMass**)。默认情况下没有引力场。

作用在一个顶点上的弹簧力，只由连接它和其他顶点的边线所施加。沿着一条边的弹簧力是弹簧刚度(**springStiffness**)乘以(顶点之间的距离与弹簧标准长度(**springLength**)之差）除以顶点之间的距离。当距离小于弹簧标准长度时，弹簧力将两个顶点分开；当距离大于弹簧标准长度时，弹簧力将两个顶点拉到一起。你可以为所有边设置默认弹簧长度(**defaultSpringLength**)以及默认弹簧刚度(**defaultSpringStiffness**)。

当两个顶点之间的距离小于一个单位时，布局将使用随机数生成器来决定力的方向。对于顶点初始位置固定的布局来说，每次布局可能产生截然不同的结果。将随机数生成器(**randomNumberGenerator**)设置为 null，以便在给定相同初始顶点位置的情况下产生可重现的结果。

该布局算法寻求具有局部最小能量的实体配置，即顶点位置作为受力点，以便于作用在每个顶点上的力之和为零。这是通过反复迭代计算每个顶点上的力、移动它们来实现的。当所有顶点的移动距离都不超过最小移动距离(**epsilonDistance)**，或迭代次数(**maxIterations**)达到最大值时，计算就会停止。

该布局不能保证节点处于最佳位置。节点通常不会相互重叠，但如果是密集的连接网络，重叠可能无法避免。

该布局使用了一个包含若干顶点(**ForceDirectedVertex)**和边(**ForceDirectedEdge)**的布局网络，顶点和边与视图中的节点和链接一一对应。

## 构造器

new go.ForceDirectedLayout(): **[ForceDirectedLayout](F:/GoJS.git/trunk/api/symbols/ForceDirectedLayout.html)**。

创建一个**ForceDirectedLayout**对象，初始时没有布局网络(network属性为null)，也没有绑定的视图(diagram属性为null)。

返回值类型：**ForceDirectedLayout。**

## 属性

### arrangementSpacing

释义为**排列间距**，数据类型为**Size**，可读写，默认值为**Size(100, 100)**。

排列空间是指构成网络的连通图之间的距离。在一个连通图中，两个任意节点之间都存在一条或多条由链接构成的通联路径。连通图之间没有链接相连。排列间距默认值为Size(100, 100)。布局开始时，首先使用排列间距分散排列多个联通图；之后使用的力导向布局算法，可能会导致连通图之间的空间大小发生变化，而且也许会发生很大的变化。

### arrangesToOrigin

释义为**移动至原点标识**，数据类型为**boolean**，可读写，默认值为**false**。

该标识表示，在执行**commitNodes()**方法时，是否移动所有节点，以至于这些节点所占的矩形区域的左上角处于原点位置(**arrangementOrigin**)。默认情况下该标识为false，表明忽略原点位置功能。该标识为true时，所有节点都将移动，包括位置固定的节点(**isFixed**属性为true)在内。

### comments

释义为**注释节点参与布局标识**，数据类型为**boolean**，可读写，默认值为**false**。

该标识表示，是否应该查找所有数据类别为”Comment”，且其锚点(注释对象)为网络中节点的节点，并且将这些注释节点作为顶点(**ForceDirectedVertex**)添加至布局网络中，参与布局。

### currentIteration

释义为**当前迭代数**，数据类型为**number**，**只读**。

该只读属性返回布局算法的当前迭代次数，只在调用**doLayout()**方法的过程中有效。

### defaultCommentElectricalCharge

释义为**注释顶点的默认电荷量**，数据类型为**number**，可读写，默认值为**5**。

该属性值是**electricalCharge()**方法的备用返回结果。

### defaultCommentSpringLength

释义为**注释顶点的默认弹簧长度**，数据类型为**number**，可读写，默认值为**10**。

该属性值是**springLength()**方法的备用返回结果。

### defaultElectricalCharge

释义为**顶点的默认电荷量**，数据类型为**number**，可读写，默认值为**150**。

该属性值是**electricalCharge()**方法的备用返回结果。

### defaultGravitationalMass

释义为**顶点的默认引力质量**，数据类型为**number**，可读写，默认值为**0**。

该属性值是**gravitationalMass()**方法的备用返回结果。

### defaultSpringLength

释义为**边线的默认弹簧长度**，数据类型为**number**，可读写，默认值为**50**。

该属性值是**springLength()**方法的备用返回结果。

### defaultSpringStiffness

释义为**边线的默认弹簧刚度**，数据类型为**number**，可读写，默认值为**0.05**。

该属性值是**springStiffness()**方法的备用返回结果。

### epsilonDistance

释义为**顶点的最小移动距离**，数据类型为**number**，可读写，默认值为**1**。

在迭代计算布局的过程中，当某次迭代完成，且至少有一个顶点在本次迭代中移动距离超过该属性值时，将进行下一次迭代计算，否则将停止布局（表明布局已合理）。该值必须大于0。

### infinityDistance

释义为**顶点电斥力的最大影响距离**，数据类型为**number**，可读写，默认值为**1000**。

当两个顶点之间的距离超过该属性值时，电斥力将忽略不计。该值必须大于1。

### maxIterations

释义为**布局最大迭代次数**，数据类型为**number**，可读写，默认值为**100**。

该属性值是力导向布局算法的最大迭代次数。当布局算法达到最大迭代次数时，无论布局是否合理，布局都将停止。该值必须为非负数。

### moveLimit

释义为**顶点单次可移动的最大距离**，数据类型为**number**，可读写，默认值为**10**。

在单次迭代算法进行时，顶点的最大移动距离不能超过该属性值。该值必须大于1。

该属性从**1.8版本**开始启用。

### randomNumberGenerator

释义为**随机数生成器**，数据类型为**object**，可读写，默认值为**Math**。

该属性值的默认值是**Math**，这将调用Math.random()生成随机数。把该属性值设为null时，布局将使用一个内部的伪随机数生成器的实例作为该属性的新值，该伪随机数生成器的返回值是可复验的、有固定规律的，此时在视图数据不变的情况下，力导向布局的结果每次都相同，不再有随机性。

该属性的新值必须为null，或者是一个object，且object带有一个名为“random”、无参数的、返回值在0(包括0)到1(不包括1)之间的方法。

该属性从**1.5版本**开始启用。

### setsPortSpots

释义为**设置链接端点为默认点标识**，数据类型为**boolean**，可读写，默认值为**true**。

该属性值表示是否要把每个链接的起始点属性(**fromSpot**)和到达点属性(**toSpot**)设置为**Spot.Default**。该属性的设置会影响到链接的两端将要连在节点的什么部位(连在节点的哪个GraphObject上)。

## 方法

### addComments

**addComments(v: [ForceDirectedVertex](https://gojs.net/latest/api/symbols/ForceDirectedVertex.html)): void**

释义为**为目标顶点添加注释**，方法类型为**Virtual**。

查找要与参数v的节点属性v. node一起定位的任何关联对象。

当**comments**属性为 true 时, 将为布局网络中的每个顶点调用此方法。该方法的标准行为是寻找类型属性(**category**)为“Comment”、且与v. node关联的节点。默认情况下, 除非将**comments**属性设置为 true, 否则该方法不会被调用。

你可能需要重写此方法, 以便自定义如何找到任何关联的对象, 以及如何将新的顶点(**ForceDirectedVertex**)和边线(**ForceDirectedEdge**)添加到网络中以表示注释。该方法会把新注释顶点的电荷量属性(**charge**)设置为**defaultCommentElectricalCharge**，并把新注释边线的长度属性(**length**)属性设置为**defaultCommentSpringLength**。

该方法从**1.3版本**开始启用。

|  |  |  |
| --- | --- | --- |
| 参数 | v: **ForceDirectedVertex** | 目标顶点 |
| 返回值 | void | - |

### commitLayout

**commitLayout(): void**

释义为**提交布局**，方法类型为**Protected**、**Override**。

在布局算法完成后，会自动调用此方法，设置链接的起始端点和到达端点，根据顶点位置为对应的节点设置视图上的位置，以及确定链接路由。

该方法会调用**commitNodes()**方法和**commitLinks()**方法，其中后者只在**isRouting**属性为true时才会调用。你不应调用此方法——它是一种“protected virtual”方法。请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | void | - |
| 返回值 | void | - |

### commitLinks

**commitLinks(): void**

释义为**提交所有链接**，方法类型为**Protected**、**Virtual**。

该方法由**commitLayout()**方法调用。它只有在**isRouting**属性为true时才会被调用。另请参阅**commitNodes()**方法。请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | void | - |
| 返回值 | void | - |

### commitNodes

**commitNodes(): void**

释义为**提交所有节点**，方法类型为**Protected**、**Virtual**。

提交所有的节点位置。

该方法由**commitLayout()**方法调用。另请参阅**commitLinks()**方法。请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | void | - |
| 返回值 | void | - |

### createNetwork

**createNetwork(): [ForceDirectedNetwork](https://gojs.net/latest/api/symbols/ForceDirectedNetwork.html)**

释义为**创建布局网路**，方法类型为**Override**。

创建一个新的布局网络(**LayoutNetwork**)**，**其中包含若干顶点(**ForceDirectedVertex**)和边线(**ForceDirectedEdge**)。

|  |  |  |
| --- | --- | --- |
| 参数 | void | - |
| 返回值 | **ForceDirectedNetwork** | 一个新的布局网络 |

### doLayout

**doLayout(coll: [Diagram](https://gojs.net/latest/api/symbols/Diagram.html) | [Group](https://gojs.net/latest/api/symbols/Group.html) | [Iterable](https://gojs.net/latest/api/symbols/Iterable.html)<[Part](https://gojs.net/latest/api/symbols/Part.html)>): void**

释义为**执行布局**，方法类型为**Override**。

执行力导向布局。

如果**network**属性为空，该方法将调用**makeNetwork()**方法创建一个包含指定部件集合coll的新布局网络(**LayoutNetwork**)。生成网络时，任何反身边线(边线两端都在同一个顶点上)都将被删除，因为它们应该被忽略。

对于每个顶点，该方法调用**electricalCharge()**方法并将返回结果赋值给顶点的电荷量属性(**charge**)，调用**gravitationalMass()**方法并将返回结果赋值给顶点的质量属性(**mass**)。

对于每条边线，该方法调用**[springStiffness](https://gojs.net/latest/api/symbols/ForceDirectedLayout.html" \l "springStiffness)()**方法并将返回结果赋值给边线的刚度属性(**stiffness**)，调用**springLength()**方法并将返回结果赋值给边线的弹簧长度属性(**length**)。

然后，布局将开始迭代计算，根据每个顶点上的力更新其位置, 直到迭代次数达到**maxIterations**, 或者直到没有顶点的移动距离超过**epsilonDistance**。

最后，该方法将调用**updateParts()**方法，根据顶点位置提交节点位置。**updateParts()**方法在一个事务中调用**commitLayout()**方法。

|  |  |  |
| --- | --- | --- |
| 参数 | coll: **[Diagram](https://gojs.net/latest/api/symbols/Diagram.html)**| **[Group](https://gojs.net/latest/api/symbols/Group.html)**| **[Iterable](https://gojs.net/latest/api/symbols/Iterable.html)**<**[Part](https://gojs.net/latest/api/symbols/Part.html)**> | 一个视图、或者组织、或者部件集合 |
| 返回值 | void | - |

### electricalCharge

**electricalCharge(v: [ForceDirectedVertex](https://gojs.net/latest/api/symbols/ForceDirectedVertex.html)): number**

释义为**获取目标顶点的电荷量**，方法类型为**Virtual**。

返回目标顶点v的电荷量。如果v.charge是一个数字，则返回v.charge，否则返回**defaultElectricalCharge**。

两个顶点之间的电斥力与它们之间的距离的平方成反比。如果顶点之间的距离超过**infinityDistance**，则认为它们彼此之间没有电斥力影响。

请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | v: **ForceDirectedVertex** | 目标顶点 |
| 返回值 | **number** | 目标顶点的电荷量 |

### electricalFieldX

**electricalFieldX(x: number, y: number): number**

释义为**获取作用在目标位置的X轴方向电场**，方法类型为**Virtual**。

返回处于指定位置的顶点所受到的X轴方向的电场。默认情况下, 任何位置都没有电场。

我们习惯于定义一个外部电场，它作用在一个与顶点电荷无关的点上。顶点L在X轴方向的受力公式为：



请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | x: **number** | 目标位置的X轴坐标 |
| y: **number** | 目标位置的Y轴坐标 |
| 返回值 | **number** | 作用在目标位置的X轴方向的电场。默认实现方法返回0。 |

### electricalFieldY

**electricalFieldY(x: number, y: number): number**

释义为**获取作用在目标位置的Y轴方向电场**，方法类型为**Virtual**。

返回处于指定位置的顶点所受到的Y轴方向的电场。默认情况下, 任何位置都没有电场。

我们习惯于定义一个外部电场，它作用在一个与顶点电荷无关的点上。顶点L在X轴方向的受力公式为：



请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | x: **number** | 目标位置的X轴坐标 |
| y: **number** | 目标位置的Y轴坐标 |
| 返回值 | **number** | 作用在目标位置的X轴方向的电场。默认实现方法返回0。 |

### gravitationalMass

**gravitationalMass(v: [ForceDirectedVertex](https://gojs.net/latest/api/symbols/ForceDirectedVertex.html)): number**

释义为**获取目标顶点的质量**，方法类型为**Virtual**。

返回目标顶点v的质量。如果v.mass是一个数字，则返回v.mass，否则返回**defaultGravitationalMass**。请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | v: **ForceDirectedVertex** | 目标顶点 |
| 返回值 | **number** | 目标顶点的质量 |

### isFixed

**isFixed(v: [ForceDirectedVertex](https://gojs.net/latest/api/symbols/ForceDirectedVertex.html)): boolean**

释义为**判定目标顶点是否为固定顶点**，方法类型为**Virtual**。

该方法返回true时，表示目标顶点不该受布局算法影响而移动，但是它依然会对临近且相连的其他顶点产生影响。默认实现方法返回v.isFixed。请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | v: **ForceDirectedVertex** | 目标顶点 |
| 返回值 | **boolean** | 返回true表示目标顶点不该受布局算法影响而移动 |

### moveFixedVertex

**moveFixedVertex(v: [ForceDirectedVertex](https://gojs.net/latest/api/symbols/ForceDirectedVertex.html)): void**

释义为**移动目标顶点**，方法类型为**Virtual**。

可以用来移动固定顶点。在每次算法迭代中，为每个顶点调用一次该方法。默认情况下该方法不执行任何操作。

|  |  |  |
| --- | --- | --- |
| 参数 | v: **ForceDirectedVertex** | 目标顶点 |
| 返回值 | void | - |

### springLength

**springLength(e: [ForceDirectedEdge](https://gojs.net/latest/api/symbols/ForceDirectedEdge.html)): number**

释义为**获取目标边线的弹簧长度**，方法类型为**Virtual**。

返回目标边线的弹簧长度。如果e.length是一个数字，则返回e.length，否则返回**defaultSpringLength**。连接两个顶点的边线E的受力公式为：



请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | e: **ForceDirectedEdge** | 目标边线 |
| 返回值 | **number** | 目标边线的弹簧长度。 |

### springStiffness

**springStiffness(e: [ForceDirectedEdge](https://gojs.net/latest/api/symbols/ForceDirectedEdge.html)): number**

释义为**获取目标边线的弹簧刚度**，方法类型为**Virtual**。

返回目标边线的弹簧刚度。如果e.stiffness是一个数字，则返回e.stiffness，否则返回**defaultSpringStiffness**。

由一条边线相连的两顶点之间的弹簧力，与弹簧长度和距离之间的差异按距离线性成比。当距离大于长度时, 力就会拉近顶点之间的距离。当距离小于长度时, 力将它们分开。连接两个顶点的边线E的受力公式为：



请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

|  |  |  |
| --- | --- | --- |
| 参数 | e: **ForceDirectedEdge** | 目标边线 |
| 返回值 | **number** | 目标边线的弹簧刚度。 |

# Layout(布局)

**更新至version 2.0.8**

继承关系：

* **Layout**
* **[GridLayout](https://gojs.net/latest/api/symbols/GridLayout.html)**
* **[CircularLayout](https://gojs.net/latest/api/symbols/CircularLayout.html)**
* **[ForceDirectedLayout](https://gojs.net/latest/api/symbols/ForceDirectedLayout.html)**
* **[LayeredDigraphLayout](https://gojs.net/latest/api/symbols/LayeredDigraphLayout.html)**
* **[TreeLayout](https://gojs.net/latest/api/symbols/TreeLayout.html)**
* **[PackedLayout](https://gojs.net/latest/api/symbols/PackedLayout.html)**
* **[RadialLayout](https://gojs.net/latest/api/symbols/RadialLayout.html)**
* **[SerpentineLayout](https://gojs.net/latest/api/symbols/SerpentineLayout.html)**
* **[SpiralLayout](https://gojs.net/latest/api/symbols/SpiralLayout.html)**
* **[TableLayout](https://gojs.net/latest/api/symbols/TableLayout.html)**
* **[TreeMapLayout](https://gojs.net/latest/api/symbols/TreeMapLayout.html)**

该类是所有预定义的关系图布局实现的基类。布局只会排列视图中的部件(主要是指节点(**Node**)和链接(**Link**))，它不会对面板(**Panel**)中的图形对象(**GraphObject**)进行布局。

布局类包括树形布局(**TreeLayout**)，力导向布局(**ForceDirectedLayout**)、分层有向布局(**LayeredDigraphLayout**)、环形布局(**CircularLayout**)以及网格布局(**GridLayout**)。此基类不是抽象的--事实上, 此基类的实例是于**Diagram.layout**和 **Group.layout**的默认值。

布局类的实例将是**Diagram.layout**或**Group.layout**的值。**Diagram.layout**定位视图中最顶级的节点和链接。属于组(**Group**)的节点和链接，将由**Group.layout**进行布局。在布局整个关系图之前, 关系图将自动执行所有嵌套的组布局。

如果在加载节点的模型数据时已准备了所有节点的位置信息，通常会将节点的位置属性(**location**)绑定在节点数据的某个属性上。为了避免布局的初始化将这些预设位置打乱，你可以不为**Diagram.layout**设值，或者将布局的允许初始化属性(**isInitial**)设为false。

布局将忽略那些允许被布局属性(**isLayoutPositioned**)为false或者可见属性(**visible**)为false的部件。布局也将忽略那些处于临时层(层的临时属性**isTemporary**为true)中的部件。

对部件的各种操作将使当前布局失效，进而触发重新自动布局。这些操作包括添加或删除部件，改变部件的可见性，或者改变部件的大小。你可以通过设置部件的布局条件属性(**layoutConditions**)来控制触发自动布局的失效条件。

但在部件上的操作并不是布局失效的唯一方式。设置布局的大多数属性，从而改变其行为, 也将使布局无效。替换视图布局或组织布局将自动使新布局无效。如果布局的观察口大小属性(**isViewportSized**)为true，当视图的观察口边界属性(**viewportBounds**)发生变化时，视图的布局将失效。(对于网格布局**GridLayout**来说，只有当布局的包装宽度属性**wrappingWidth**为NaN时，之前将的观察口变化才会生效。大部分布局并不关心观察口的大小。)

你可以明确地通过调用布局的**layoutDiagram()**方法，使布局失效并重新执行布局。但是我们建议您尽量不要这样做，而是根据需要执行正常的更新布局过程。

如果自动布局是第一次对模型执行布局, 则视图将首先触发名为 "InitialLayoutCompleted" 的视图事件(**DiagramEvent**)。每当视图完成自动布局时, 它都会触发名为 "LayoutCompleted" 的视图事件。

你也可以显示调用布局的**doLayout()**方法进行布局，但这并不常见，仅用于那些不属于任何视图或组织的布局实例。只有当你需要布局某节点和链接的集合, 而且这些节点和链接是视图的非顶级部件或是某组织的子图时，才应该需要它。

更复杂的布局使用由若干顶点(**LayoutVertex**)和边线(**LayoutEdge**)组成的单独的布局网络(**LayoutNetwork**)，该网络通常包含一个与节点和链接所组成图形同构的图形结构。布局的**doLayout()**方法将调用**makeNetwork()**方法，并将结果记为网络属性(**network**)。**makeNetwork()**方法将调用**createNetwork()**方法，并通过添加顶点和边线的实例来初始化布局网络，这些顶点和边线与给定的节点和链接一一对应。

当**doLayout()**方法完成工作后，它将调用**updateParts()**方法，该方法将调用**commitLayout()**方法来为节点和链接设置新的位置和路由。然后，布局网络通常会被丢弃(**network**属性设置为null)。

顶点和边线实例允许布局处理有关每个节点和链接的详细信息, 而无需实际修改这些节点和链接, 直到调用**commitLayout()**方法以实际设置节点位置和链接路由。使用 **LayoutNetwork**还允许布局使用与给定的节点和链接集合不同构的图形。这在需要使用虚拟顶点和(或)边缘来实现某些布局行为时非常有用, 或者当你想要忽略某些顶点或边时, 而无需实际修改、添加或删除视图中的节点或链接。

此布局基类的实例提供了一个基本的默认布局, 该布局将定位所有没有位置的部件 (即 部件的位置属性**location**是(NaN,NaN))。已具有位置的部件将被忽略。此基础布局不使用 **LayoutNetwork**, 因为它忽略了所有链接。

若要实现自己的自定义布局, 可以从此类继承, 也可以从其他预定义布局类之一继承。如果从这个基类继承, 则需要重写**doLayout()**方法。你可以调用部件的**move()**方法来重设部件的位置，组织(**Group**)也可以使用此方法。请阅读 "**Extensions**" 的简介页, 了解如何重写方法以及如何调用此基方法。

## 构造器

new go.Layout(): **[Layout](F:/GoJS.git/trunk/api/symbols/ForceDirectedLayout.html)**。

创建一个最小的布局，仅定位没有位置的节点。

返回值类型：**[Layout](F:/GoJS.git/trunk/api/symbols/ForceDirectedLayout.html)。**

## 属性

### arrangementOrigin

释义为**布局原点**，数据类型为**Point**，可读写，默认值为**Point(0, 0)**。

该属性值是视图在布局时应定位到的左上角。默认值为Point(0, 0)，也就是视图坐标原点。改变此属性的值会使布局无效，从而触发重新布局。在执行组织(**Group**)内的布局时，很有可能需要设置此属性值。

### boundsComputation

释义为**边界计算方法**，数据类型为function(part: **Part**, lay: **Layout**, rect: **Rect**): **Rect** | null，可读写，默认值为null。

该属性是一个方法，它用来确定与节点相对应的顶点的初始大小和在文档坐标系中的位置。该方法由**getLayoutBounds()**方法调用。此属性的默认值为 null, 在这种情况下, 顶点的大小和位置使用的是节点的**GraphObject.actualBounds**属性。改变此属性的值会使布局无效，从而触发重新布局。

该属性非空时，必须是一个接受三个参数的方法。第一个参数是将要使被布局使用其边界的部件。第二个参数是该布局。第三个参数是必须修改和返回的矩形。返回值必须在视图坐标系中。你可能会发现，通过调用**GraphObject.getDocumentBounds()**方法，很容易获取节点内部对象在文档坐标中的边界。

该属性从**2.0版本**开始启用。

### diagram

释义为**布局所属视图**，数据类型为**Diagram** | null，可读写，默认值为null。

如果该布局是**Diagram.layout**，则可以通过该属性获取布局所属视图。

如果该属性和**group**属性都不为null，该组织应该在该视图中。

### group

释义为**布局所属组织**，数据类型为**Group**| null，可读写，默认值为null。

如果该布局是**Group.layout**，则可以通过该属性获取布局所属组织。

### isInitial

释义为**允许布局初始化标识**，数据类型为boolean，可读写，默认值为true。

该属性表示此布局是否执行初始化布局。默认值为true。把该属性值设置为false会引发**isValidLayout**属性被设置为true，从而致使视图不会执行此布局。

如果你把**isInitial**和**isOngoing**属性都设置为false，则不会出现自动布局无效的情况，因为**invalidateLayout()**方法不会把**isValidLayout**属性设置为false。在这种情况下，如果想要你的节点显示，你需要明确设置或者将**Part.location**或**GraphObject.location**绑定在真正的Point值上，因为自动布局不会分配任何位置。

另一种控制布局何时失效的方法是设置**Part.isLayoutPositioned**或 **Part.layoutConditions**。

### isOngoing

释义为**允许布局失效标识**，数据类型为boolean，可读写，默认值为true。

该属性表示是否可以通过**invalidateLayout()**方法使此布局无效，从而触发重新布局。将此属性设置为 false, 以防止添加或删除部件等操作导致此布局无效。改变此属性不会使此布局无效。

如果你把**isInitial**和**isOngoing**属性都设置为false，则不会出现自动布局无效的情况，因为**invalidateLayout()**方法不会把**isValidLayout**属性设置为false。在这种情况下，如果想要你的节点显示，你需要明确设置或者将**Part.location**或**GraphObject.location**绑定在真正的Point值上，因为自动布局不会分配任何位置。

另一种控制布局何时失效的方法是设置**Part.isLayoutPositioned**或 **Part.layoutConditions**。

### isRealtime

释义为**允许实时布局标识**，数据类型为boolean | null，可读写，默认值为null。

该属性表示在一个事务结束之前，布局是否可被实时地执行。一般来说，所有无效的布局都将在事务结束时执行。该属性默认值为null，null对于**Diagram.layout**表示true，对于**Group.layout**则表示false。改变此属性不会使此布局无效。

该属性从**1.2版本**开始启用。

### isRouting

释义为**允许控制链接路由标识**，数据类型为boolean，可读写，默认值为true。

该属性表示此布局是否可以控制链接路由，将链接设置为直线、折线以及曲线等路由形状，以美化布局外观。该属性默认值为true。如果为false，则此布局将不会显式设置**Link.points**，并且在**commitLayout()**方法部署节点位置后，每个链接将会以默认路由形状绘制。改变此属性不会使此布局无效。

某些布局忽略链接, 在这种情况下, 此属性将被忽略。

该属性从**1.1版本**开始启用。

### isValidLayout

释义为**布局失效标识**，数据类型为boolean，可读写，无默认值。

该属性表示此布局是否需要被重新执行，为false时表示需要。一般不会直接设置该属性。通常布局在调用**invalidateLayout()**方法时将该属性设置为false，因为之后肯定要执行布局。

### isViewportSized

释义为**布局依赖观察口大小标识**，数据类型为boolean，可读写，默认值为false。

该属性表示此布局是否取决于**Diagram.viewportBound**的大小。该属性如果被设置为true，布局将会在视图的观察口大小改变时失效。该属性只适用于视图布局，而非组织布局，而且只有当**Diagram.autoScale**被设置为**Diagram.None**时才生效。改变此属性不会使此布局无效。

### network

释义为**布局网络**，数据类型为LayoutNetwork | null，可读写，默认值为false。

该属性表示此布局使用的布局网络 (如果有)。设置此属性不会使此布局无效。并非所有类型的布局都使用布局网络。可以调用**createNetwork()**或**makeNetwork()**来创建一个布局网络。