Grid

1. 网址：

<https://github.com/hootsuite/grid#gridlist-class>

1. 翻译：

有2个作用：

1. 不可知网格列表类，它管理虚拟矩阵中的项目列表中的二维位置。
2. 构建于GridList类之上的jQuery插件，该插件将通用项位置转换为具有拖放功能的响应DOM元素

GridList class

API

基本体

**API**

**1.**

new GridList(items, options)

var myGrid = new GridList(items, {

direction: 'horizontal',

lanes: 3

});

第一个构造函数参数（items）是用网格填充网格的数组。

Supported options:

direction - Can be 'horizontal' or 'vertical'

anes - Number of fixed rows or columns, depending on the direction.

2.

generateGrid()

myGrid.generateGrid();

使用给定项目的位置从头开始构建网格结构。如果项目缺少位置属性（X和Y），则它们将错位，可能重叠。如果希望围绕仅定义了大小属性（w和h）的项列表构建网格，并依赖库进行二维定位，用resizeGrid.

3. resizeGrid(lanes)

myGrid.resizeGrid(4);

（重新）为给定的行数/列生成网格实例中的项的位置。这种方法有两个主要用途：

1.项目第一次被二维表示。

2.项目已经具有2d位置，但是需要在不同的网格大小上表示，尽可能地保持其先前的顺序。

网格内的位置是从左到右、从上到下生成的。因此，当寻找网格内的新位置时，选择最左边列的最顶层行。

4. moveItemToPosition(item, position)

// Move item from [0, 0] to [1, 1]

var carefreeItem = myGrid.grid[0][0];

myGrid.moveItemToPosition(carefreeItem, [1, 1]);

下面是在网格内移动一个项目时发生的事情：

项目的先前位置在2D网格内被清除，

项目对象内部的位置被更新，

项目的新位置标记在2D网格内。

如果移动的项目与网格中的其他项目重叠，则处理冲突。

碰撞可以通过两种方式来解决。首先，尝试在本地解析它们，这意味着移动的项尝试与重叠的项交换位置。这是首选的公平贸易。如果不能解决这个问题，在交换之后，网格内部仍然有冲突，整个网格将重新生成，从移动的项目固定在新位置开始。在后一种情况下，移动物品周围和右侧的所有物品可能会稍微改变它们的位置。

5.

resizeItem(item, size)

// Resize item from position [0, 0] to span over 3 columns

var growthItem = myGrid.grid[0, 0];

myGrid.resizeItem(growthItem, {w: 3});

console.log(growthItem.w); // will output "3"

myGrid.resizing(growthItem, {h: 2});

console.log(growthItem.h); // will output "2"

调整项目的大小与移动其位置非常相似，也就是说网格单元将重新填充，冲突将在之后处理。参见[moveItemToPosition](https://github.com/hootsuite/grid" \l "moveitemtopositionitem-position)。

**Primitives**

1. Item

该项目是GRIDLIST的构建块，是一个简单的JavaScript对象。网格表的主要功能是在两个维度上定位这些项目。这给我们带来了一个项目的组成：W和H的大小，X和Y的位置。例如:

{w: 3, h: 1, x: 0, y: 1}

注意，x和y（列和行）是网格内的抽象坐标，它们是从0开始的整数值。当然，w和h（宽度和高度）在同一坐标系中占用空间，这揭示了网格的最小单元：单元。例如，可以说上面的特色项目占用了三个网格单元。

1. gridList.items

网格列表实例围绕项目数组工作。项目数组是类构造函数的第一个参数，在.items属性下始终可见。这里有一个列表，其中网格上有三个1x1个项目，其中一列有三行：

[{w: 1, h: 1, x: 0, y: 0},

{w: 1, h: 1, x: 0, y: 1},

{w: 1, h: 1, x: 0, y: 2}]

1. gridList.grid

看到JavaScript不支持多维数组，GridList中的2d网格由一个列数组表示，每个数组条目包含另一个数组，每个数组条目包含每个行的单元格。单元格只是一个指向它的项目的指针，或者如果没有一个项目位于该单元的位置，则为空引用。请注意，更多的单元格可以指向相同的项引用，因为项占据W\*H单元格。这里是一个网格伪表示：

| **col1** | **col2** | **col3** | **col4** |
| --- | --- | --- | --- |
| 1 | 2 |  |  |
| 1 | 3 | 4 | 4 |
| 1 |  | 4 | 4 |

将网格置于二维数据结构中，我们可以直接通过将网格中覆盖的任何单元格作为目标来获取项目引用。例如。

myGrid.grid[1][0] // reference to item #2// ，一次表示一个数

myGrid.grid[1][1] // reference to item #3

myGrid.grid[2][1] // reference to item #4

myGrid.grid[3][2] // still reference to item #4

此网格将由这些项生成：//?

[{w: 1, h: 3, x: 0, y: 0},

{w: 1, h: 1, x: 1, y: 0},

{w: 1, h: 1, x: 1, y: 1},

{w: 2, h: 2, x: 2, y: 1}]

//以表格左上为（0,0）原点，先看坐标，再向右下方延伸，大小是看w和h

且相同的编号是一块

4.

$.fn.gridList

$('.my-list').gridList({

direction: 'horizontal',

lanes: 3

});

jQuery插件有两个主要功能：

1.将网格列表放在DOM元素列表的顶部。列表项预期具有data-w和data-h属性，并且可选地具有data-x和data-y（如果它们的位置先前已经生成并持久化）

2.拖放功能

该函数采用可选的参数，这些选项将在构造它们时传递给拖曳体。

$('.my-list').gridList({lanes: 3}, {handle: '.title'});

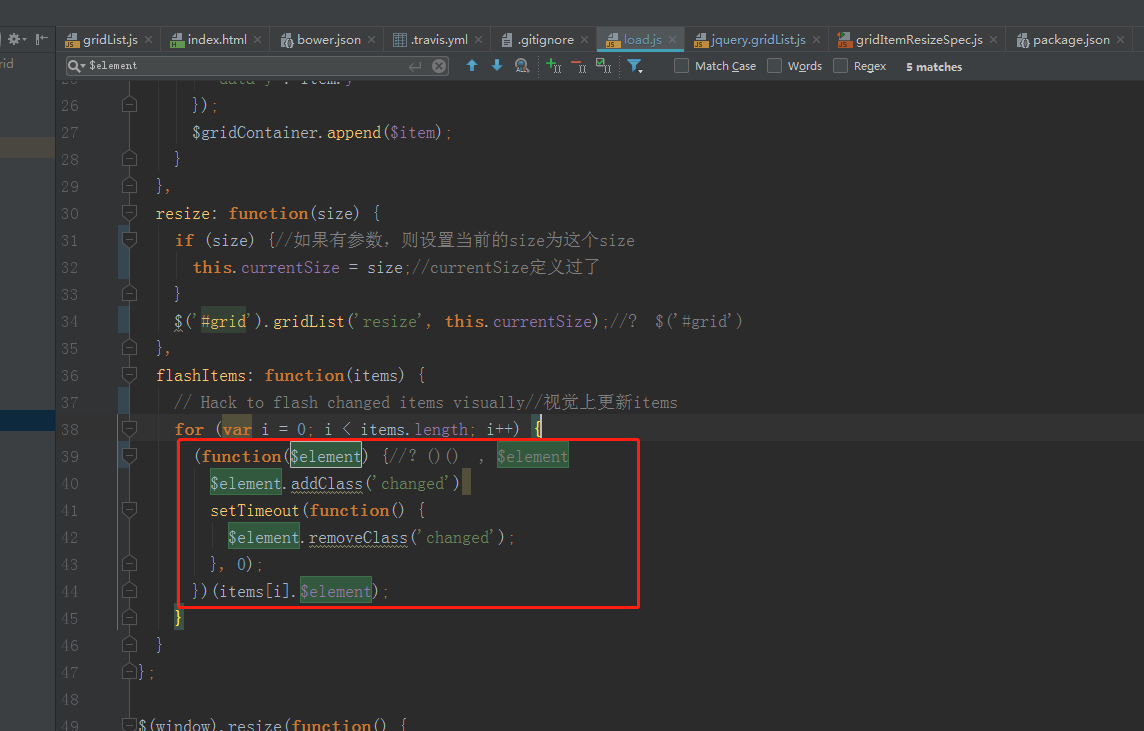
有关所有可用选项的详细信息，请参阅jQuery UI可拖动API。

呈现的列表响应于其父容器，这意味着item的宽度和高度是基于容器高度除以网格行的数量来计算的。

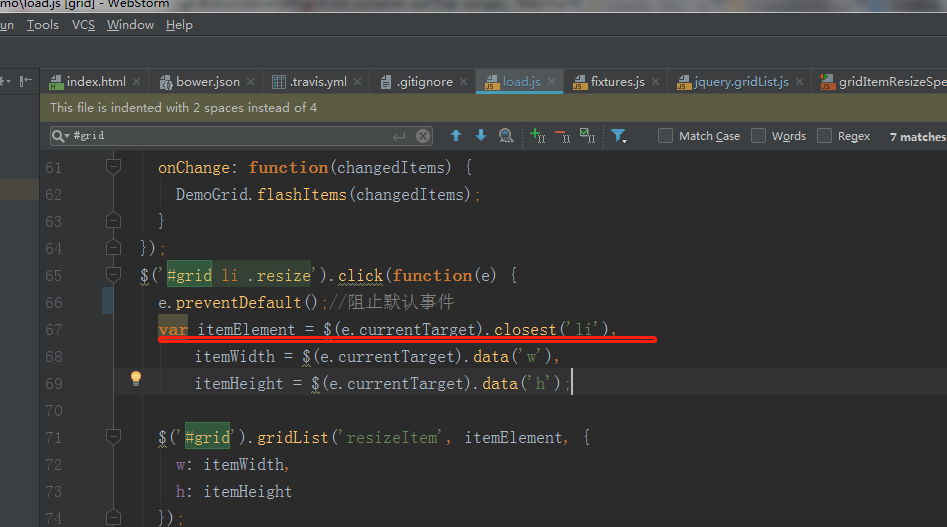
FAQ: Why not gridster?

代码的 问题：

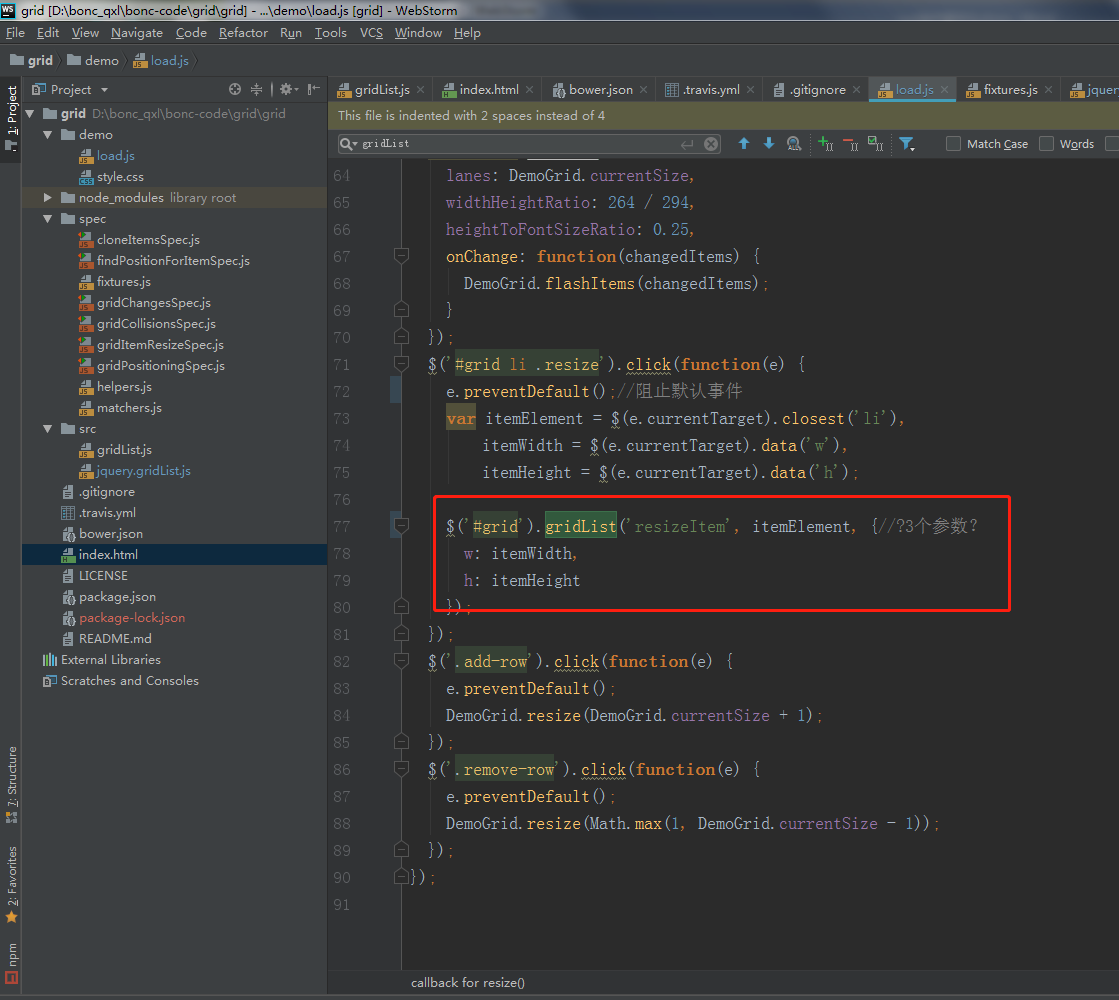
1.



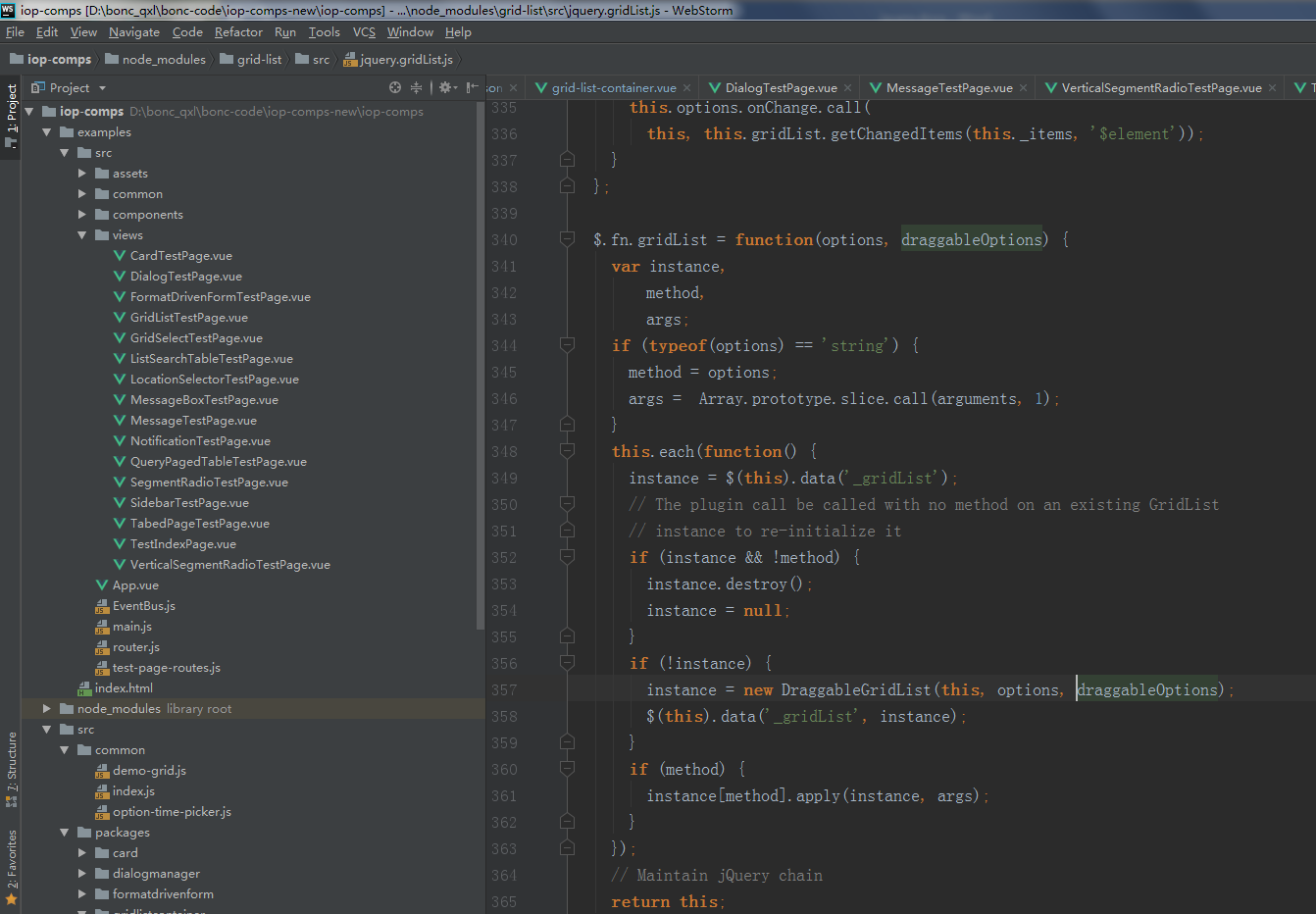
1. 注意用法



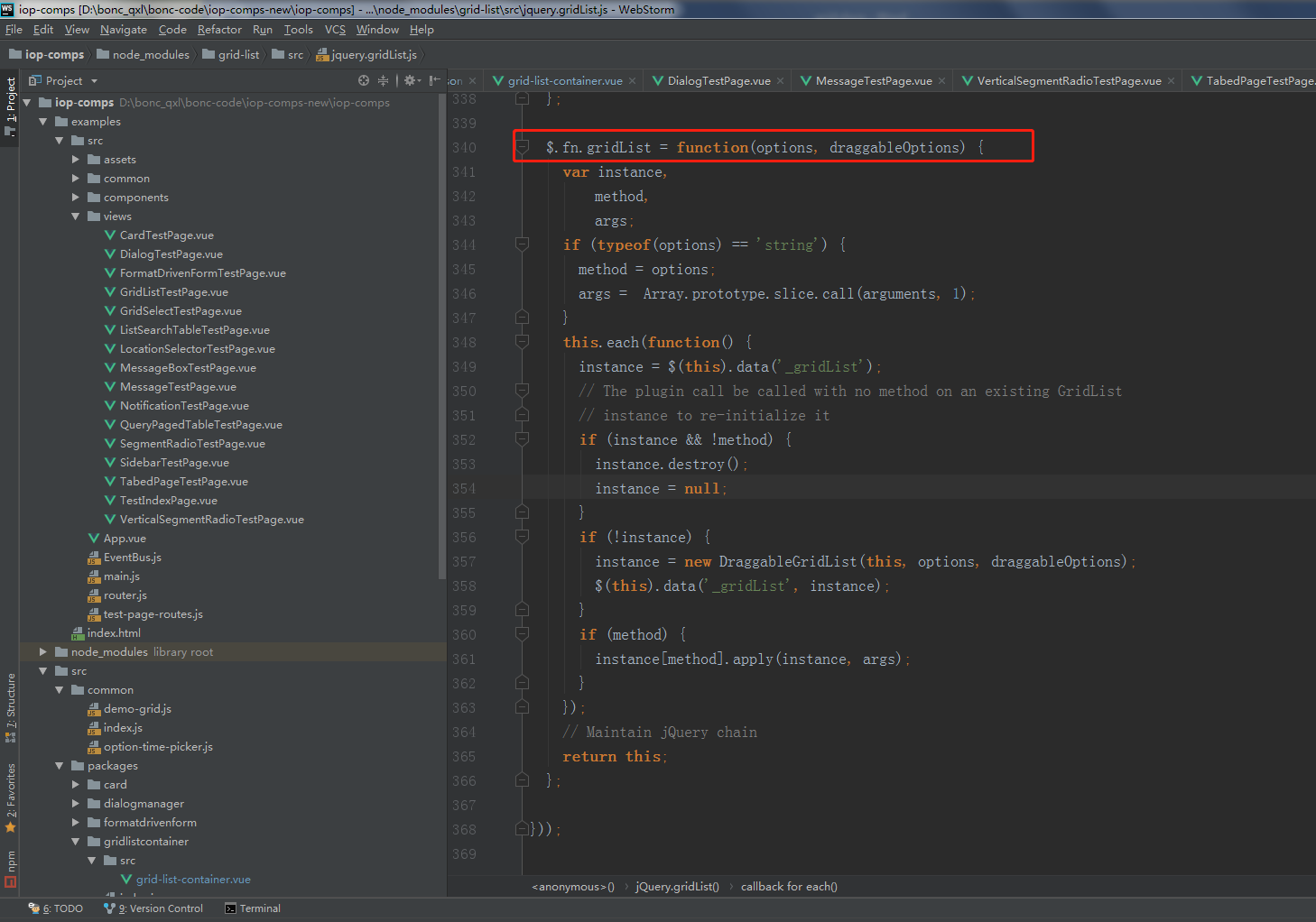
1. 这个为啥是3个参数,这个很好

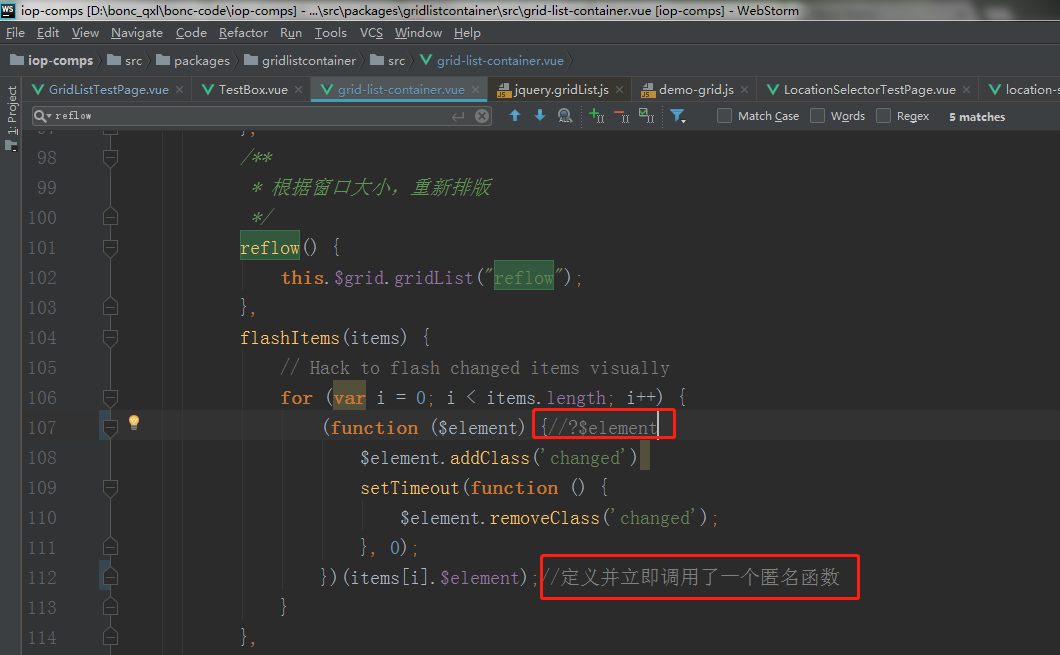


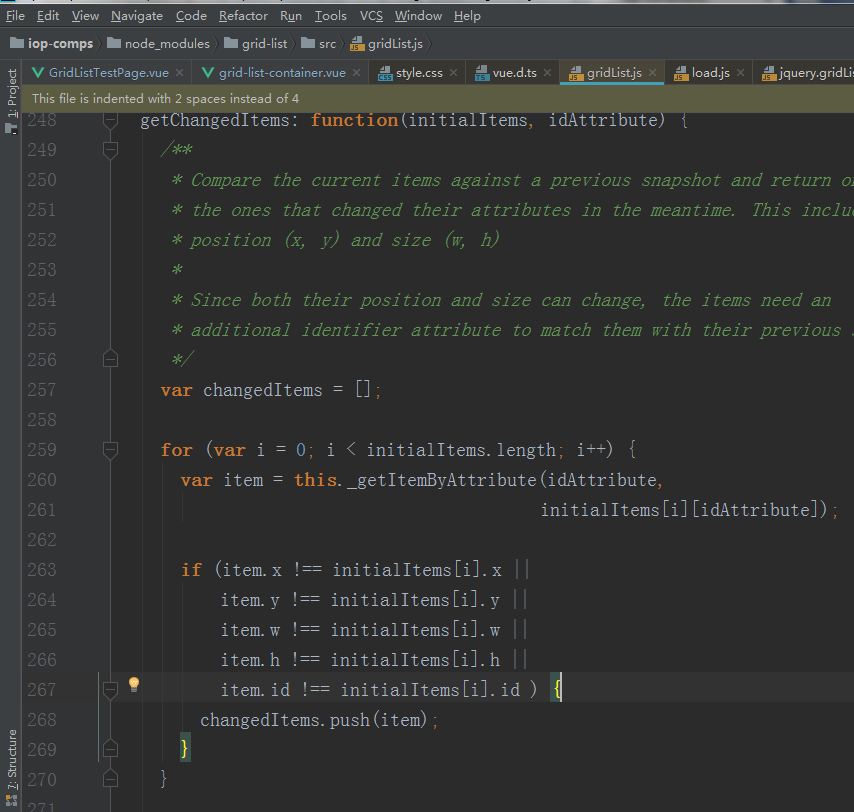
答：原因是jquery的机制，后面的参数是传给那个函数的（那个字符串resizeItem），可以看gridList()的实现：

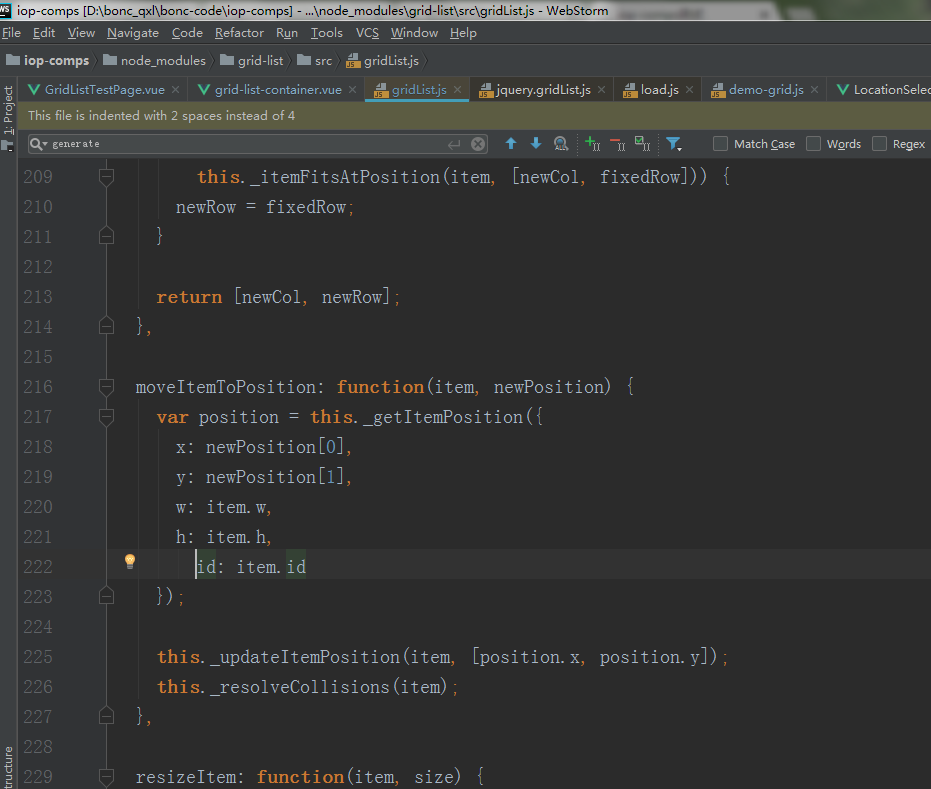


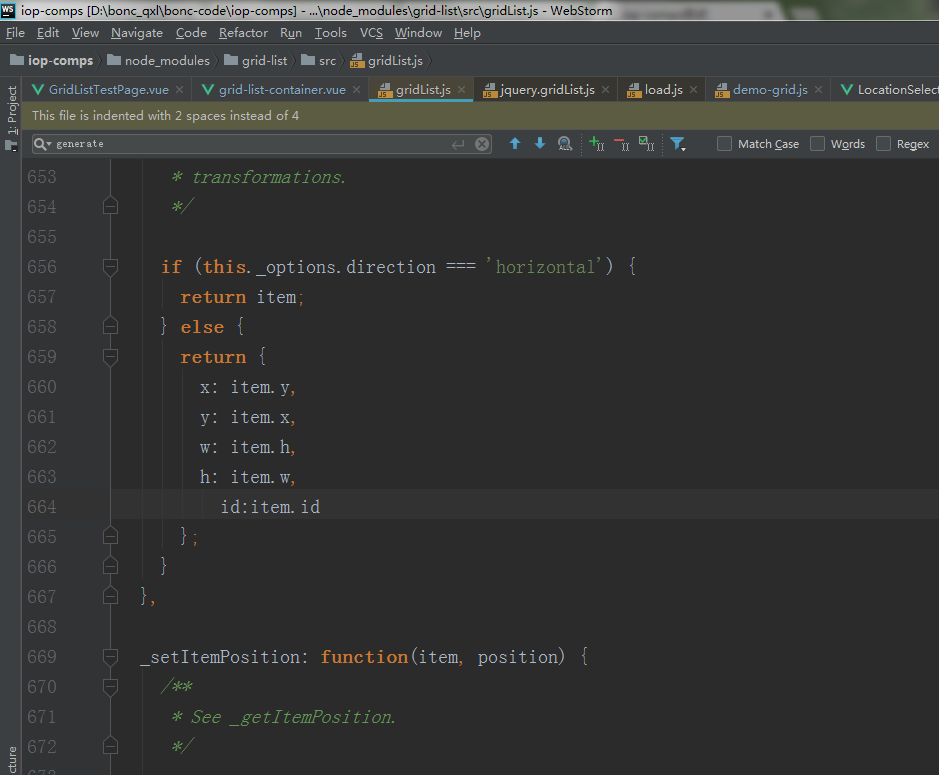
1. jquery强相关，不太懂这个代码：



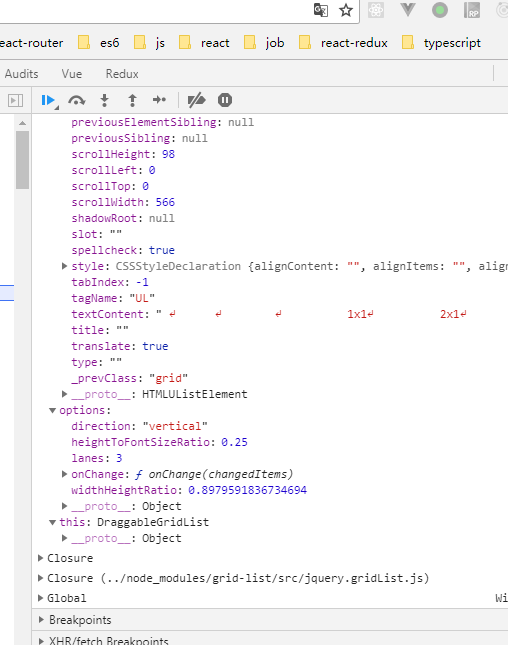
1. 这里不懂：  
   
2. 改了这些地方：



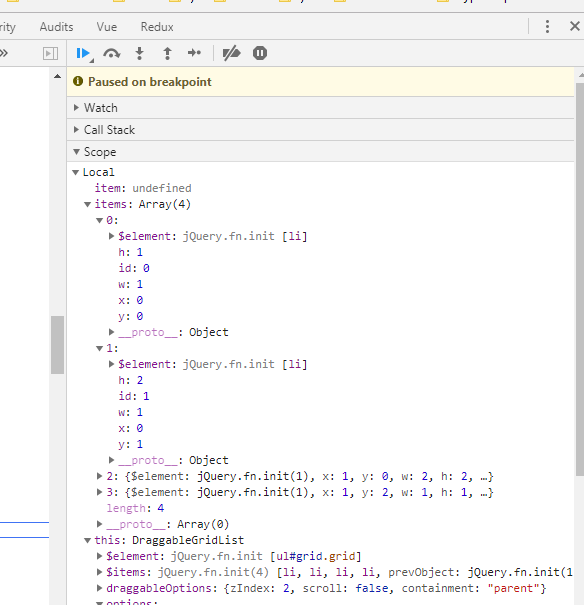


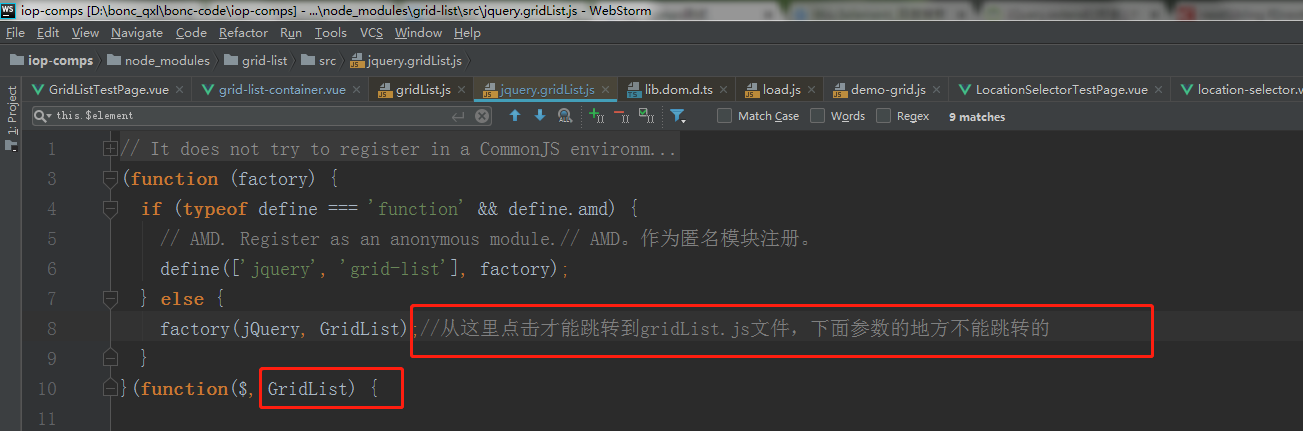


1. Options

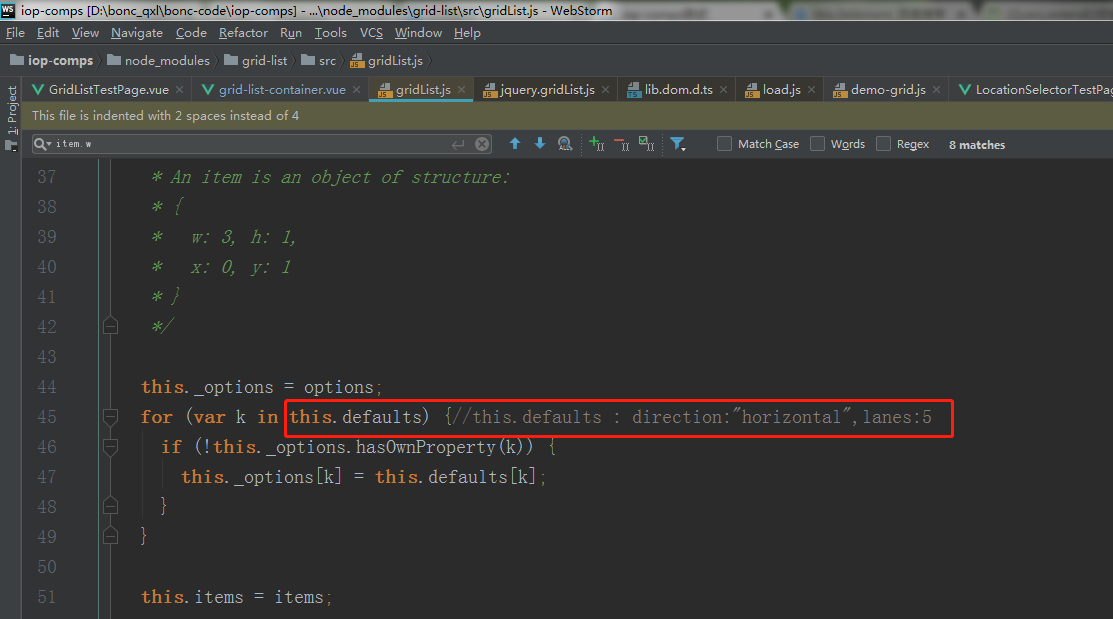


Items:

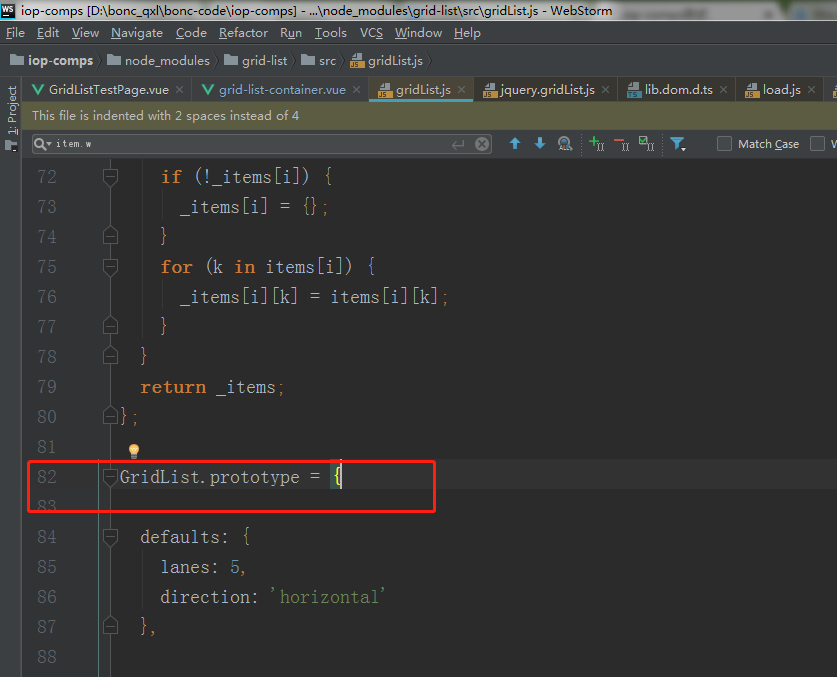




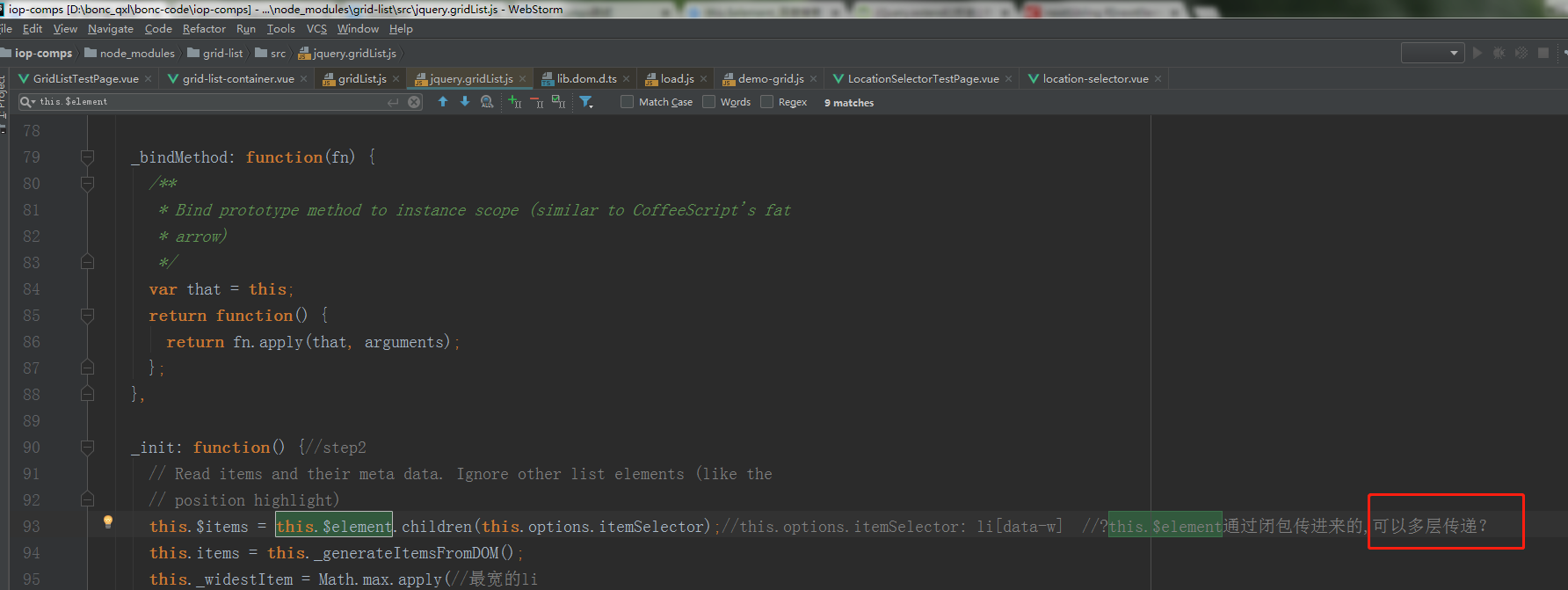
1. 这个：



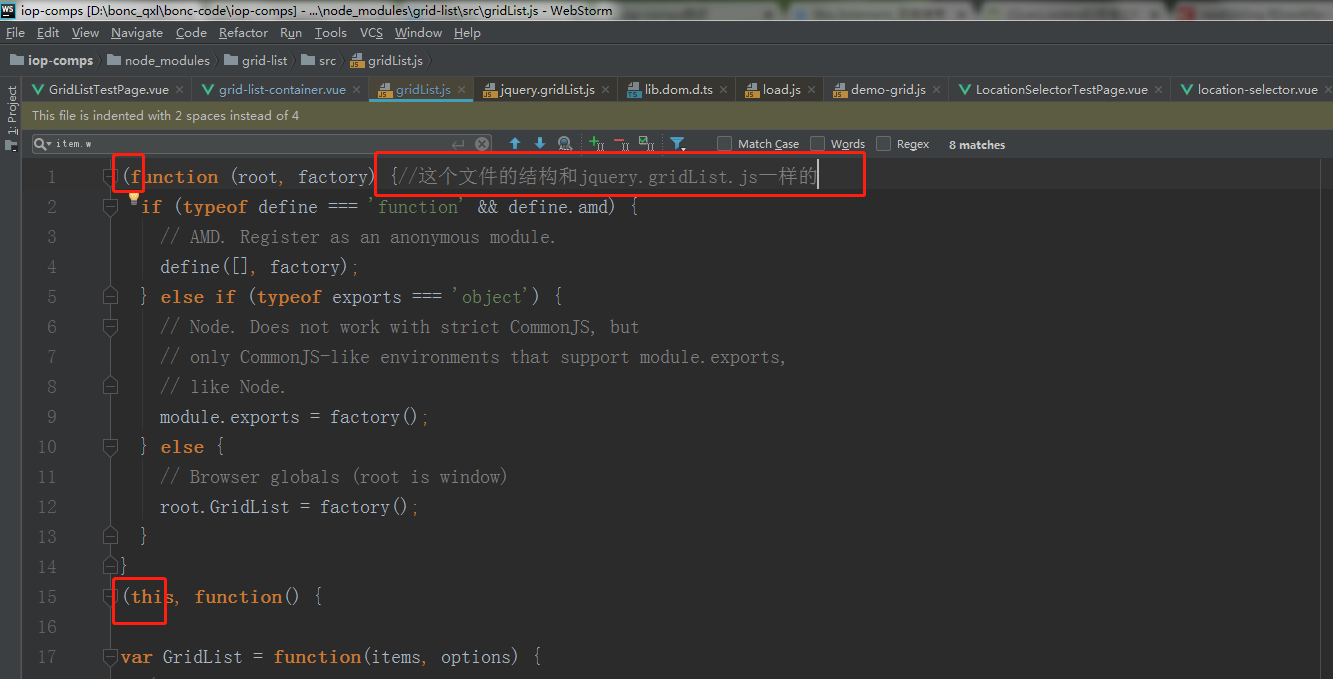
在这里：



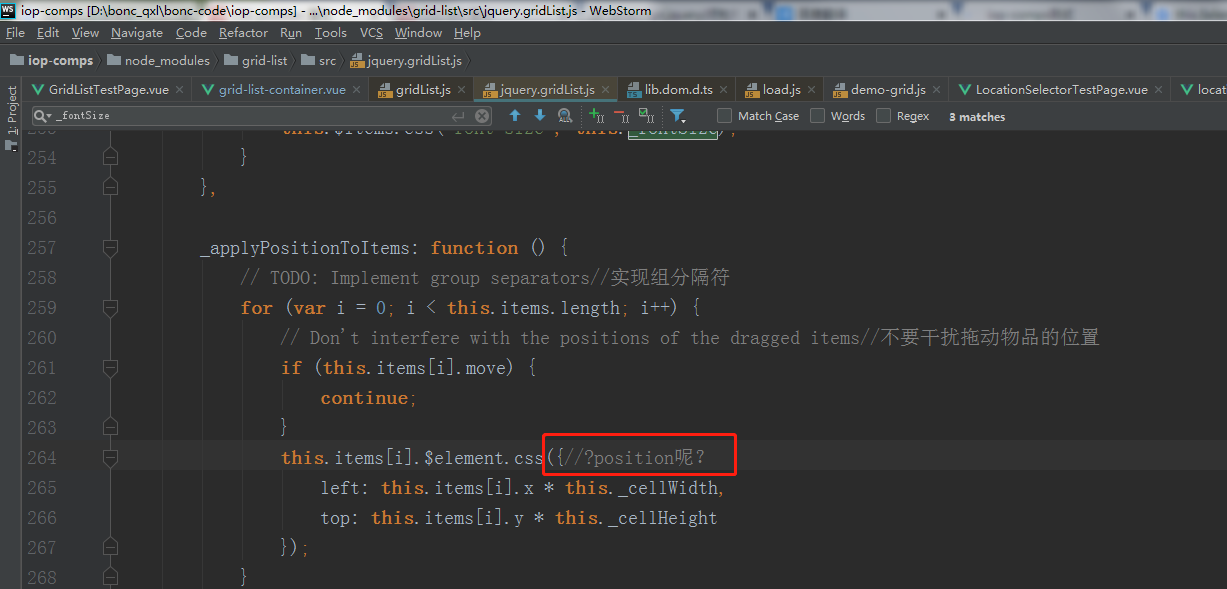
1. 问题



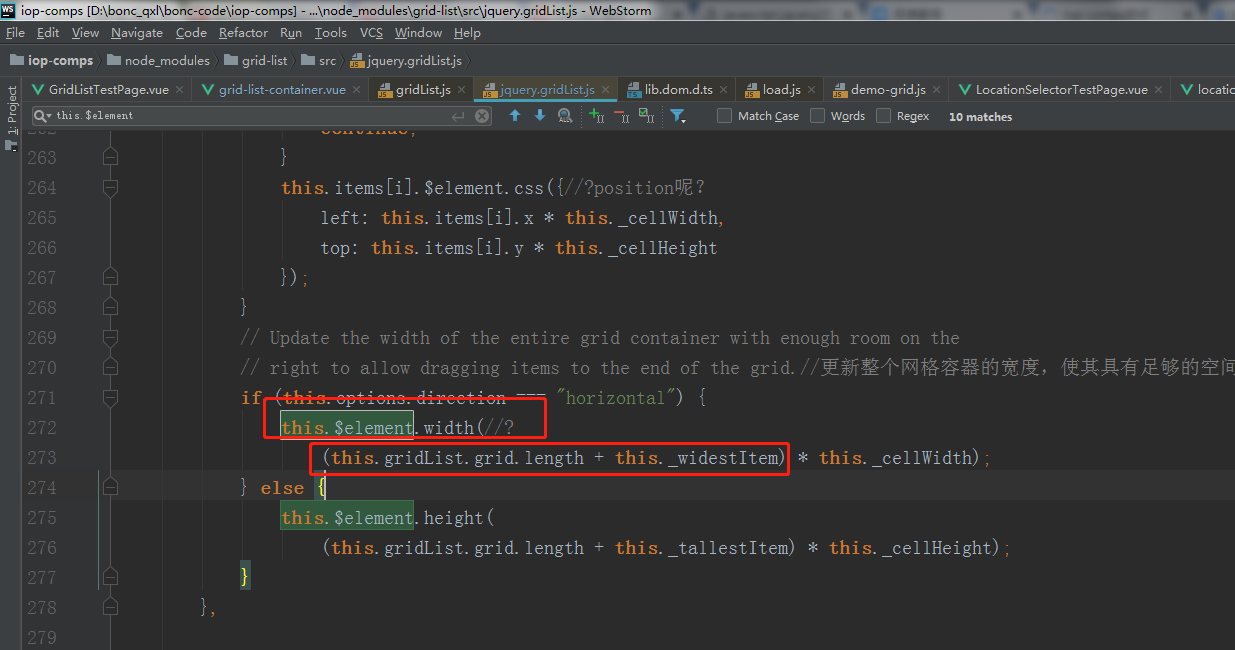
1. 这个是啥意思？模块？两个文件都看一下



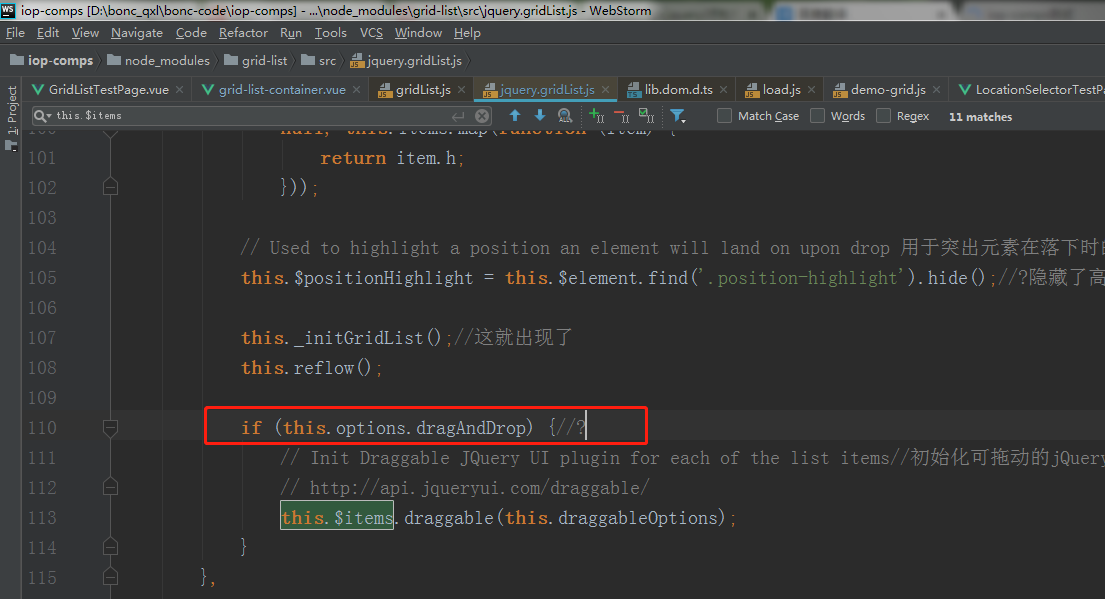
1. 问题



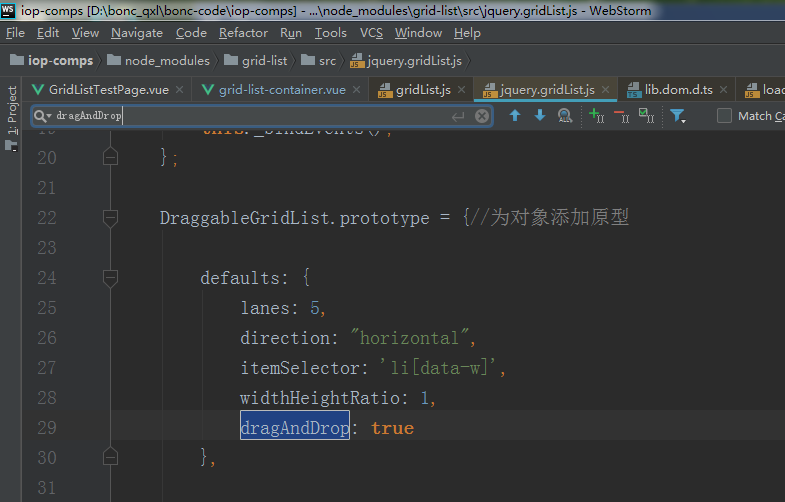
1. 问题



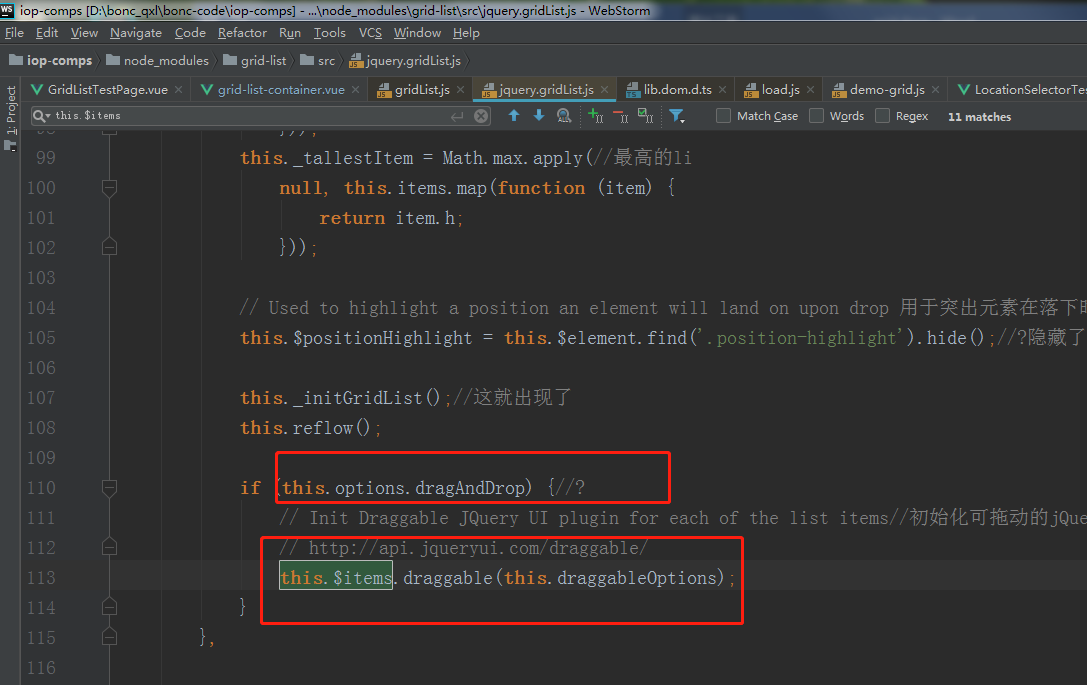
Options什么时候有这个属性的？



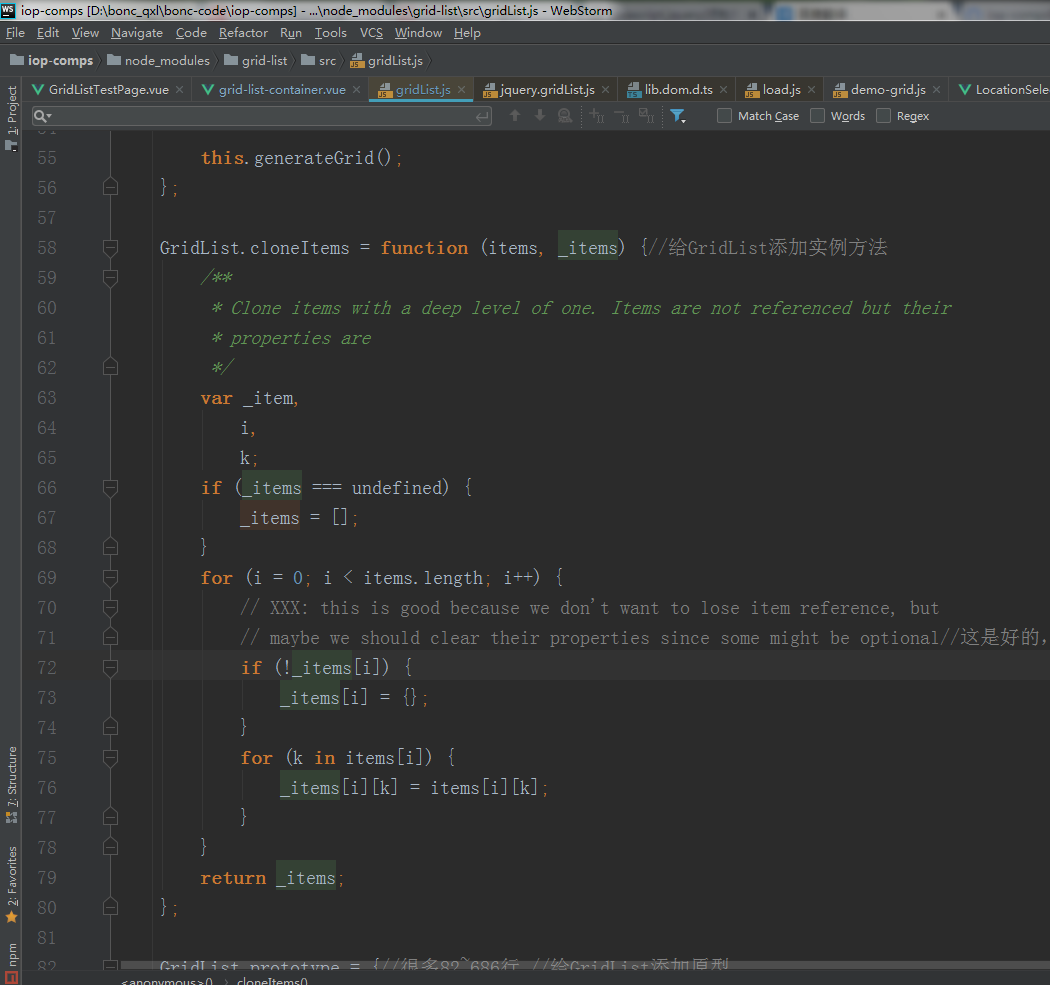
只有这里有啊：



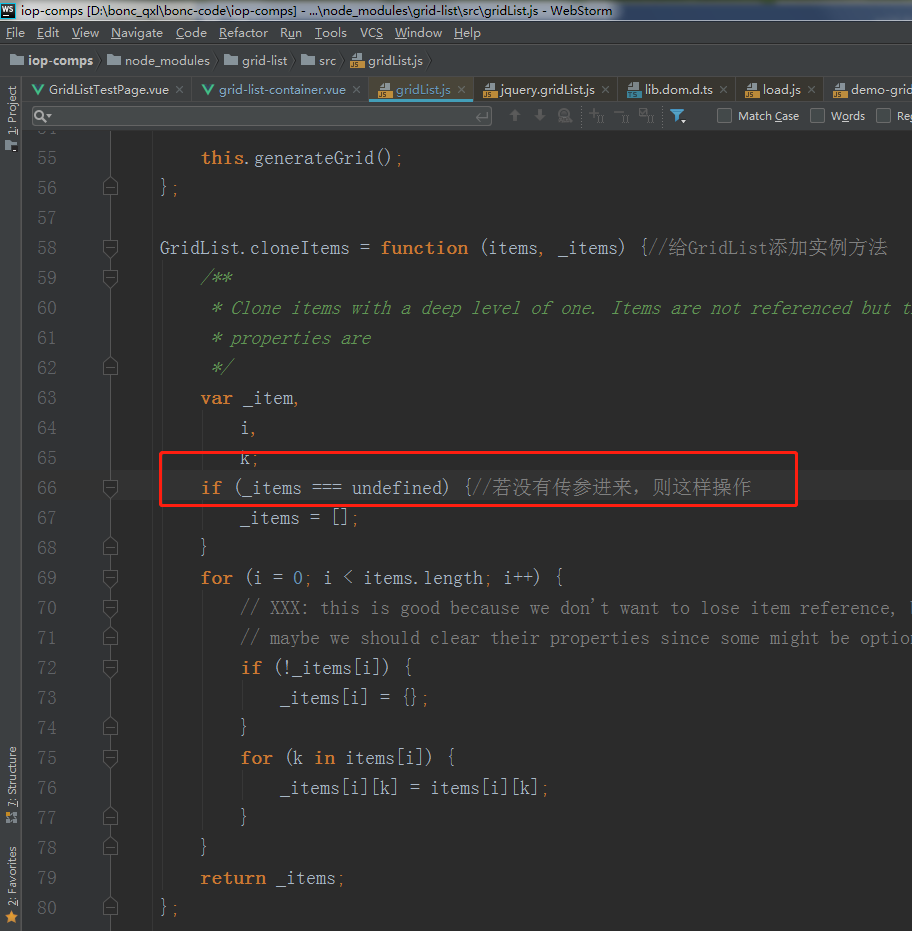
哪来的：



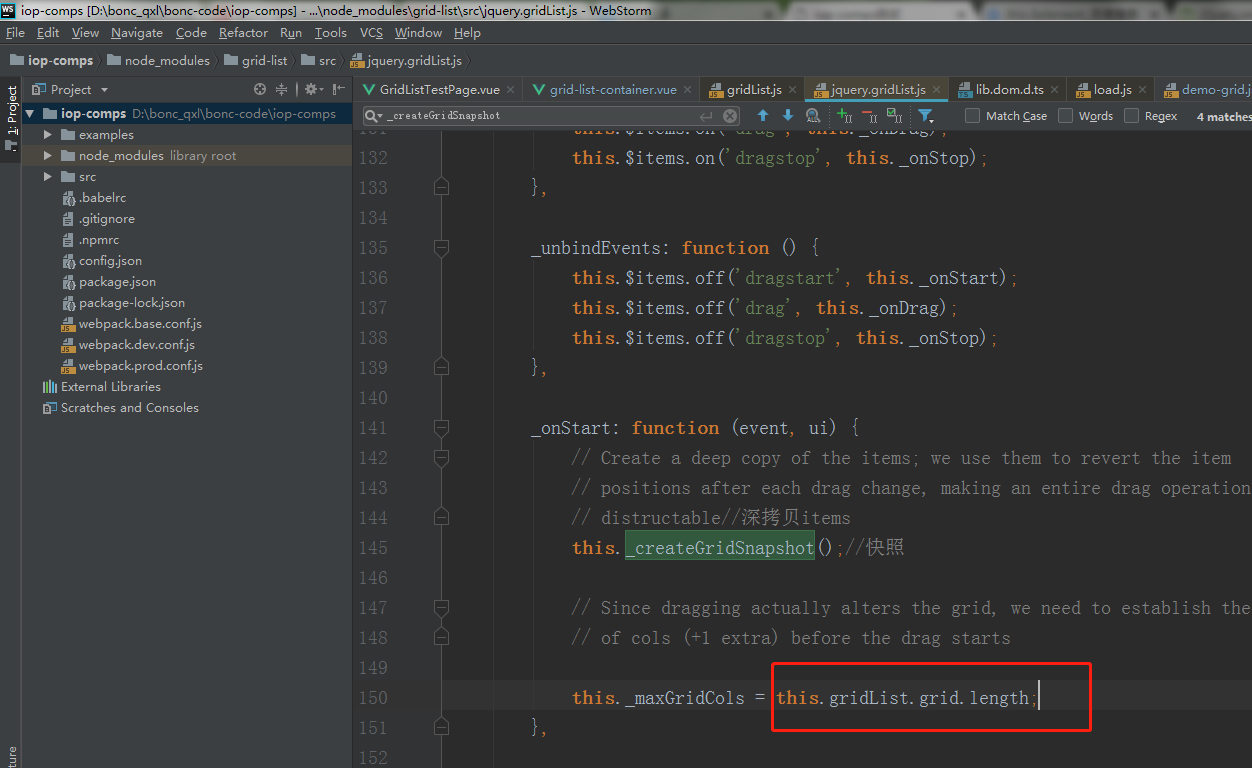
不懂这个深拷贝：



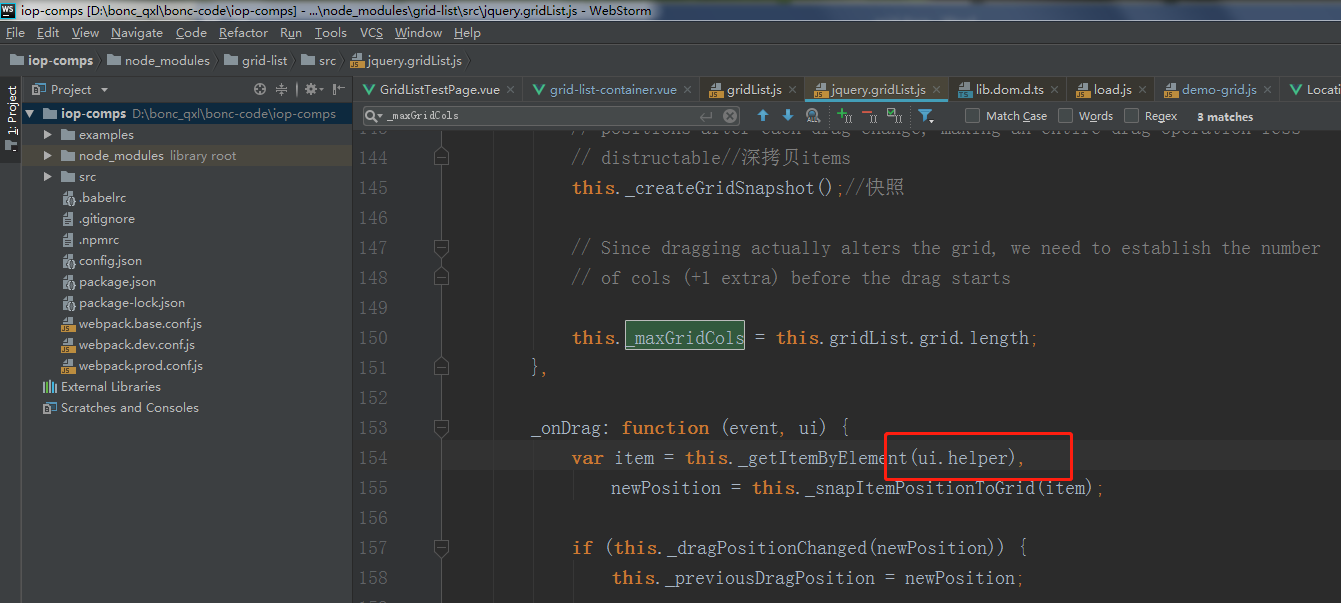
注意：判断是否传参了的方式：



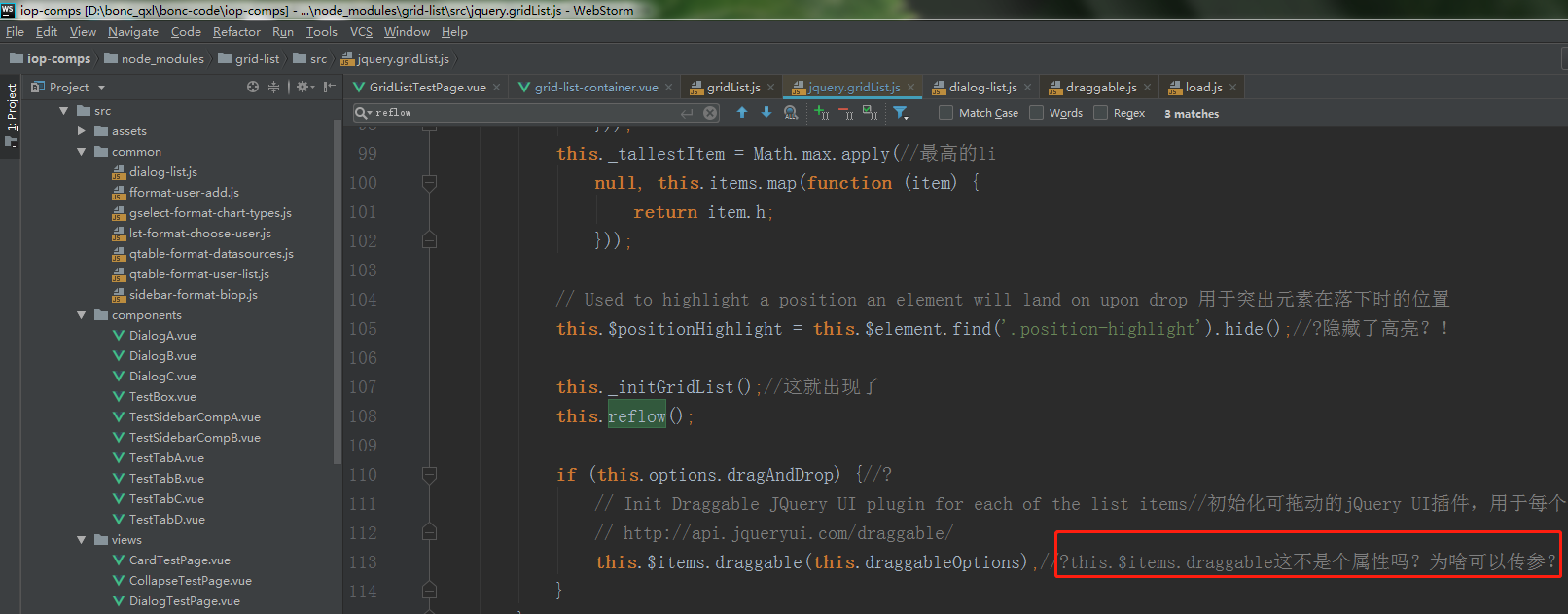
到底是啥？



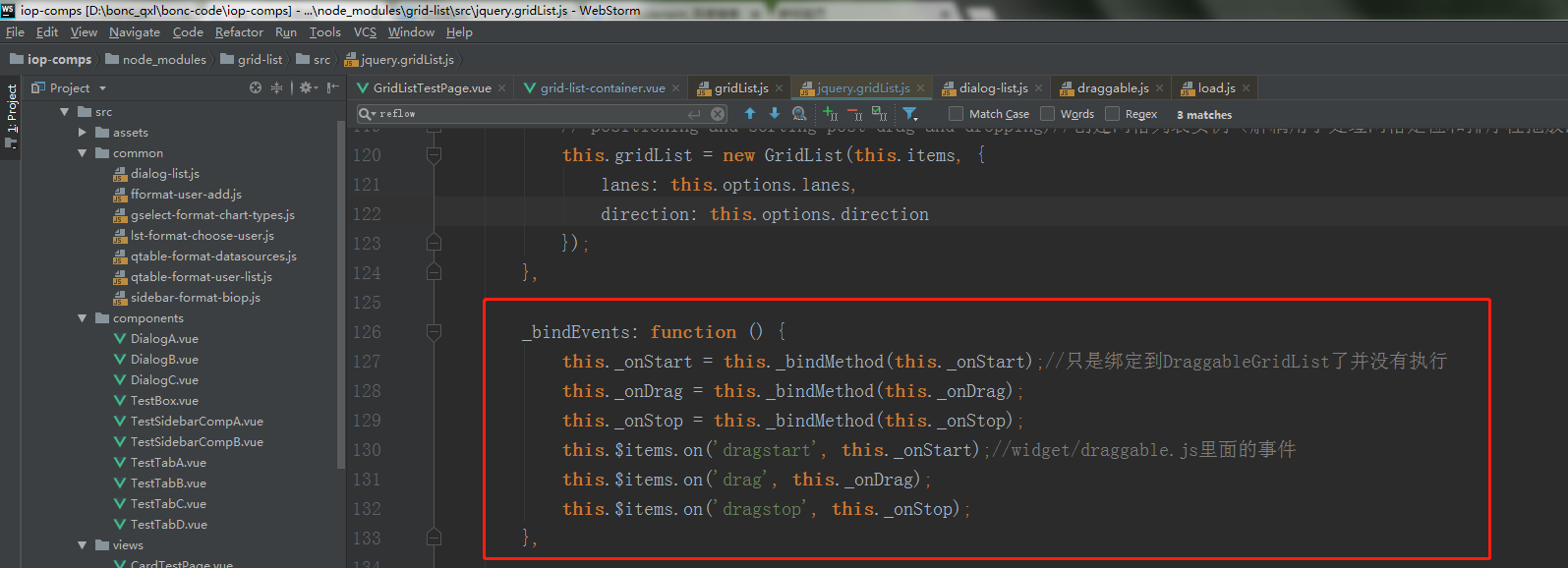
是啥？



为啥？



看不懂：



注意看：

