常规配置：

### 1.固定行列位置

fixedRowsTop:行数 //固定顶部多少行不能垂直滚动

fixedColumnsLeft:列数 //固定左侧多少列不能水平滚动

### 2.拖拽行头或列头改变行或列的大小

manualColumnResize:true/false//当值为true时，允许拖动，当为false时禁止拖动

manualRowResize:true/false//当值为true时，允许拖动，当为false时禁止拖动

### 3.延伸列的宽度

stretchH：last/all/none       //last:延伸最后一列,all:延伸所有列,none默认不延伸。

### 4.手动固定某一列

manualColumnFreeze: true/false

//当值为true时，选中某 一列，右键菜单会出现freeze this column选项，该选项的作用是固定这一列不可水平滚动，并会将这一列移动到非固定列的前面。再次右键菜单会出现unfreeze this column，意思是取消该列的固定，并将其还原到初始位置。

### 5.拖动行或列到某一行或列之后

manualColumnMove：true/false 当值为true时，列可拖拽移动到指定列

manualRowMove:true/false 当值为true时，行可拖拽至指定行

当属性的值为true时，行头或列头可以被拖拽移动，移动后该行或列将会被移动到指定位置，原先该行或列的后面部分自动上移或后退。移动的时候鼠标需选中行线或列线才行。

### 6.设置当前行或列的样式

currentRowClassName：当前行样式的名称

currentColClassName：当前列样式的名称

### 7.行分组或列分组

groups:[{cols:[0,2]},{cols:[3,5]},{rows:[0,4]},{rows:[5,7]}]

上面的例子介绍了4个分组，第0-2列为一组，第3-5列为第二组，第0-4行为一组，第5-7列为第二组。分组可在行头或列头看见，分组可被展开或隐藏。

### 8.允许排序

columnSorting：true/false/对象 //当值为true时，表示启用排序插件

当值为true时，排序插件的使用可通过点击列头文字实现。当值为false时表示禁用排序。当值为对象时，该对象包含两个属性：column：列数。sortOrder：true/false，升序或降序，true为升序排列。当用对象启动插件后可用如下方式禁用插件：

hot.updateSettings({

    columnSorting:false

});

排序的使用也可已直接调用sort()方法实现。如下操作：

if(hot.sortingEnabled){

    hot.sort(行数，true/false); //true为升序，false为降序

}

### 9.显示行头列头

colHeaders:true/fals/数组 //当值为true时显示列头，当值为数组时，列头为数组的值

rowHeaders：true/false/数组 //当值为true时显示行头，当值为数组时，行头为数组的值

### 10.数据显示

data：[[第一行数据],[第二行数据],...[第n行数据]]/对象数组

获取数据的方法：hot.getData()。

加载数据的方法：hot.loadData(data)。

当不需要显示某一列的时候可用如下格式设置：

columns：[

   {data:0},

   {data:2}

]

这里就不显示第二列数据，只有第1、3列数据

### 11.右键菜单展示

contextMenu：true/false/自定义数组 //当值为true时，启用右键菜单，为false时禁用

### 12.自适应列大小

autoColumnSize：true/false //当值为true且列宽未设置时，自适应列大小

### 13.minCols：最小列数

    minRows：最小行数

    minSpareCols：最小列空间，不足则添加空列

    maxCols：最大列数

    maxRows：最大行数

    minSpareRows：最小行空间，不足则添加空行

### 14.observeChanges：true/false //当值为true时，启用observeChanges插件

### 15.colWidths:[列宽1，列宽2，...]/列宽值

例如：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/mafan121/article/details/46050049)

1. var hot = new Handsontable(container, {
2. data: data,
3. observeChanges: true,
4. colHeaders: true,
5. rowHeaders: true,
6. colWidths: 70,
7. contextMenu: false,
8. manualRowResize: true,
9. manualColumnResize: true,
10. minSpareRows: 30,
11. cells: function(row, col, prop) {//单元格渲染
12. this.renderer = myRenderer;
13. },
14. mergeCells: true
15. });

### 16.自定义边框设置，可以进行初始化配置，如下：

customBorders:[{range:{from:{row:行数,col:列数},to:{row:行数,col:列数},上下左右设置}]

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/mafan121/article/details/46050049)

1. hot = Handsontable(container, {
2. data: Handsontable.helper.createSpreadsheetData(200, 20),
3. rowHeaders: true,
4. fixedColumnsLeft: 2,
5. fixedRowsTop: 2,
6. colHeaders: true,
7. customBorders: [
8. {
9. range: {//多个单元格
10. from: {//起始位置
11. row: 1,
12. col: 1
13. },
14. to: {
15. row: 3,
16. col: 4
17. }
18. },
19. top: {//结束位置
20. width: 2,
21. color: '#5292F7'
22. },
23. left: {
24. width: 2,
25. color: 'orange'
26. },
27. bottom: {
28. width: 2,
29. color: 'red'
30. },
31. right: {
32. width: 2,
33. color: 'magenta'
34. }
35. },
36. {//单一单元格
37. row: 2,
38. col: 2,
39. left: {
40. width: 2,
41. color: 'red'
42. },
43. right: {
44. width: 1,
45. color: 'green'
46. }
47. }]
48. });

也可以声明customBorder:true，表示允许自定义单元格边框。

### 17.单元格合并可以进行初始化配置，如下：

mergeCells: [{row: 起始行数, col: 起始列数, rowspan: 合并行数, colspan:合并列数 },...]，

也可以先声明单元格允许合并，mergeCells:true，再利用合并方法操作。

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/mafan121/article/details/46050049)

1. hot = new Handsontable(container, {
2. data: data,
3. observeChanges: true,
4. colHeaders: true,
5. rowHeaders: true,
6. colWidths: 70,
7. contextMenu: false,
8. manualRowResize: true,
9. manualColumnResize: true,
10. //  minSpareRows: 30,
11. cells: function(row, col, prop) {
12. this.renderer = myRenderer;
13. },
14. mergeCells: true,

自定义合并：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/mafan121/article/details/46050049)

1. if(key == "merge") {
2. if(hot.mergeCells.mergedCellInfoCollection.length > 0) {
3. for(var k=0; k<hot.mergeCells.mergedCellInfoCollection.length; k++) {
4. if(hot.mergeCells.mergedCellInfoCollection[k].row == row &&
5. hot.mergeCells.mergedCellInfoCollection[k].col == col) {
6. hot.mergeCells.mergedCellInfoCollection.splice(k,1);
7. return;
8. }else{
9. hot.mergeCells.mergedCellInfoCollection.push({"row": row, "col": col,
10. "rowspan": rowspan, "colspan": colspan});
11. break;
12. }
13. }
14. } else {
15. hot.mergeCells.mergedCellInfoCollection.push({"row": row, "col": col, "rowspan": rowspan, "colspan": colspan});
16. }

注意：hansontable将合并的单元格单独拿出来放到了mergeCells数组的mergedCellInfoCollection集合中，所以对合并的操作也是增删集合中的对象。并重新渲染。

LAA001180 100 & Par Driver CB006D-00 237,621 28,000 30,000 18,000 30,000 20,000 14,129