0 - 官网

<https://github.com/ag-grid/ag-grid>

1 - 学习

<https://www.ag-grid.com/angular-grid/>

yarn add ag-grid-community ag-grid-angular

import { AgGridModule } from 'ag-grid-angular';

@NgModule({

imports: [

AgGridModule.withComponents([])

],

})

export class AppModule {}

下一步是添加ag-Grid样式-

C:\yutao\software\frontend-work\angular.json：

"node\_modules/ag-grid-community/dist/styles/ag-grid.css",

"node\_modules/ag-grid-community/dist/styles/ag-theme-alpine.css",

网格有几个不同的主题。选择一个与您的项目设计相匹配的项目。您可以使用Sass变量进一步自定义它，我们将在以后的文章中进一步介绍该技术。

export class AppComponent {

title = 'my-app';

columnDefs = [

{ field: 'make' },

{ field: 'model' },

{ field: 'price'}

];

rowData = [

{ make: 'Toyota', model: 'Celica', price: 35000 },

{ make: 'Ford', model: 'Mondeo', price: 32000 },

{ make: 'Porsche', model: 'Boxter', price: 72000 }

];

}

网格的两个基本配置属性-[列定义](https://www.ag-grid.com/javascript-grid-column-definitions/)（columnDefs）和数据（rowData）

<ag-grid-angular

style="width: 500px; height: 500px;"

class="ag-theme-alpine"

[rowData]="rowData"

[columnDefs]="columnDefs"

>

</ag-grid-angular>

这是ag-grid组件定义，具有两个属性绑定-rowData和columnDefs。该组件还接受标准DOMstyle和class。我们将类设置为ag-theme-alpine，它定义了网格主题。您可能已经注意到，CSS类与我们之前导入的CSS文件的名称匹配。

启用排序和过滤

columnDefs = [

{ field: 'make', sortable: true, filter: true },

{ field: 'model', sortable: true, filter: true },

{ field: 'price', sortable: true, filter: true }

];

获取远程数据

export class AppComponent implements OnInit {

columnDefs = [

{ field: 'make', sortable: true, filter: true },

{ field: 'model', sortable: true, filter: true },

{ field: 'price', sortable: true, filter: true }

];

}

上面的代码将srowData从硬编码数组转换为Observable。为了使网格能够使用它，我们需要向该属性添加一个异步管道：

<ag-grid-angular

style="width: 500px; height: 500px;"

class="ag-theme-alpine"

[rowData]="rowData | async"

[columnDefs]="columnDefs"

>

</ag-grid-angular>

启用选择

，我们需要允许用户从网格中选择某些行，并将其标记为系统中已标记的行。

我们将标志切换状态和持久性留给后端团队。在我们这边，我们应该启用选择，然后获取选定的记录，并将它们与API调用一起传递给远程服务端点。

export class AppComponent implements OnInit {

columnDefs = [

{ field: 'make', sortable: true, filter: true, checkboxSelection: true },

{ field: 'model', sortable: true, filter: true },

{ field: 'price', sortable: true, filter: true }

];

}

接下来，启用多行选择，以便用户可以选择多行：

<ag-grid-angular

style="width: 500px; height: 500px;"

class="ag-theme-alpine"

[rowData]="rowData | async"

[columnDefs]="columnDefs"

rowSelection="multiple"

>

</ag-grid-angular>

通过不绑定属性值，我们在这里采取了一些捷径。如果不使用[]，则赋值会将属性值作为字符串传递，这对于我们的目的是很好的。

大！现在，第一列包含一个复选框，单击该复选框可以选择该行。我们唯一需要添加的就是一个按钮，它可以获取选定的数据并将其发送到服务器。为此，我们将使用 ag-Grid API-我们将通过组件实例对其进行访问。要访问组件实例，我们需要将#agGrid ID添加到我们的组件中。

<button (click)="getSelectedRows()">Get Selected Rows</button>

<ag-grid-angular

#agGrid

style="width: 500px; height: 500px;"

class="ag-theme-alpine"

[rowData]="rowData | async"

[columnDefs]="columnDefs"

rowSelection="multiple"

>

</ag-grid-angular>

现在，让实例在我们的组件中可访问：

export class AppComponent implements OnInit {

@ViewChild('agGrid') agGrid: AgGridAngular;

columnDefs = [

{ field: 'make', sortable: true, filter: true, checkboxSelection: true },

{ field: 'model', sortable: true, filter: true },

{ field: 'price', sortable: true, filter: true }

];}

我们唯一需要添加的就是getSelectedRows组件的方法。

export class AppComponent implements OnInit {

@ViewChild('agGrid') agGrid: AgGridAngular;

columnDefs = [

{ field: 'make', sortable: true, filter: true, checkboxSelection: true },

{ field: 'model', sortable: true, filter: true },

{ field: 'price', sortable: true, filter: true }

];

getSelectedRows() {

const selectedNodes = this.agGrid.api.getSelectedNodes();

const selectedData = selectedNodes.map(node => node.data );

const selectedDataStringPresentation = selectedData.map(node => node.make + ' ' + node.model).join(', ');

alert(`Selected nodes: ${selectedDataStringPresentation}`);

}

}

自定义主题外观

我们要做的最后一件事是通过修改主题的一些参数来更改网格的外观。

打开style.scss并添加一些参数：

@import "../node\_modules/ag-grid-community/src/styles/ag-grid.scss";

@import "../node\_modules/ag-grid-community/src/styles/ag-theme-alpine/sass/ag-theme-alpine-mixin.scss";

.ag-theme-alpine {

@include ag-theme-alpine((

odd-row-background-color: #CFD8DC

));

}

您可以在此处找到主题参数的完整列表。

如果一切都配置正确，则网格的第二行将变暗。恭喜你！现在，您现在知道可以随意调整网格外观了-还有几十个Sass变量，可用于控制字体系列和大小，边框颜色，标题背景颜色，甚至单元格和列中的间距。主题文档部分中提供了完整的主题参数列表。

Grid Size

为了让网格自动调整它的高度以适应行，设置网格属性domLayout='autoHeight'。

当显示大量行时，不要使用网格自动高度。

DOM布局  
有三个DOM布局值，网格可以有'normal'， 'autoHeight'和'print'。它们的用途如下:  
normal:如果没有指定，这是默认值。网格适合您提供的div的宽度和高度，并向两个方向滚动。  
autoHeight:网格的高度被设置为适合行数，因此网格不提供垂直滚动条。网格像往常一样水平滚动。  
打印:没有使用滚动条，网格渲染所有的行和列。这种布局在印刷中得到了解释。

用父容器调整大小

我们可以通过使用网格API特性动态地对屏幕变化做出反应。在本节中，我们将介绍一些根据屏幕大小变化来调整网格大小和显示/隐藏列的推荐方法。

1 Flexbox容器内  
默认情况下，网格运行一个计时器来监视其容器大小并相应地调整UI的大小。这可能会干扰display: flex set元素的默认行为。简单的解决方法是在网格元素的父元素中添加overflow: hidden。

2 在CSS网格容器内  
默认情况下，网格会监视其容器大小，并相应地调整UI的大小。这可能会干扰display: grid set元素的默认行为。简单的解决方法是在grid元素的父元素中添加overflow: hidden。

3 动态调整大小与水平滚动  
实现响应式网格的最快方法是将包含div的网格设置为百分比。通过这个简单的更改，网格将根据div大小自动调整大小，而无法放入视口的列将通过滚动条被隐藏并向右可用。

动态调整大小，不需要水平滚动  
有时，您希望将不适合当前视图的列完全隐藏起来，没有水平滚动条。  
要实现这一点，请确定网格的宽度并计算出该空间可以容纳多少列，隐藏不适合的列，并根据gridSizeChanged事件触发不断更新，如下一个示例所示。

动态垂直调整  
有时网格的垂直高度大于它所在的行数。您可以动态设置行高来填充可用的高度，如下面的示例所示:

调整列的大小以适应  
调用api.sizeColumnsToFit()使当前可见的列适合屏幕。列将缩放(增加或缩小)以适应可用的宽度

如果您不希望在自动调整大小中包含特定的列，那么设置列定义suppressSizeToFit=true。例如，如果您希望第一列保持固定宽度，而其他所有列填充表的宽度，那么这是很有帮助的。

Auto-Size列  
就像Excel一样，每一列都可以通过双击标题右侧而不是拖动来“自动调整大小”。当您这样做时，网格将计算出适合列中单元格内容的最佳宽度。

关于自动调整大小列，请注意以下几点:

1 网格通过只考虑虚拟渲染的行来计算出最佳宽度。例如，如果您的网格有10,000行，但由于行的虚拟化只呈现了50行，那么将只考虑这50行来计算要显示的宽度。呈现的行是通过水平滚动加上一个小缓冲区(默认缓冲区大小为20)可以在屏幕上看到的所有行。

2 自动调整列大小将查看屏幕上呈现的单元格，并根据所看到的内容计算出宽度。它无法看到由于列虚拟化而未呈现的列。因此，不可能自动调整屏幕上不可见的列的大小。

列虚拟化是一种网格用于呈现大量列的技术，它只呈现由于水平滚动位置而可见的列，从而降低了性能。例如，如果水平滚动只显示10列，那么网格可以有1000列，只有10列呈现。

为了解决这个问题，你可以通过设置grid属性suppressColumnVirtualisation=true来关闭列虚拟化。您可以选择是想让列虚拟化工作还是使用屏幕外的列自动调整大小

默认情况下，网格也会调整列的大小以适应标题。如果不希望在自动大小计算中包含标头，则设置网格属性skipHeaderOnAutoSize=true。

Autosize列API  
还可以使用以下网格API方法自动调整列的大小:  
autosizeccolumn (colKey, skipHeader):自动调整一个列。  
autosizecolcolumns (colKeys, skipHeader):自动调整许多列的大小。  
autoSizeAllColumns(colKey, skipHeader)：自动调整所有列的大小。  
如果skipHeader=true，在计算列宽度时将不包括标题。  
在计算列宽度时从不考虑列组。

## Column Flex 通常需要一个或多个列填充网格中的整个可用空间。对于这个场景，可以使用flex配置。一些列可以用常规的宽度配置来设置，而其他的列可以用flex配置。

伸缩调整的工作原理是将网格中的剩余空间按伸缩值的比例划分到所有伸缩列中。例如，假设网格的总宽度为450px，它有三列:第一列宽度为150;第二个是flex: 1;第三个是flex: 2。第一列宽度为150px，剩余300px。使用flex: 2的列的大小是使用flex: 1的两倍。最终的尺寸是:150px 100px 200px。

flex配置不能在同一列中使用宽度配置。如果你需要为一个列提供最小宽度，你应该使用flex和minWidth配置。Flex也将maxWidth考虑在内。

如果您使用flex通过API或拖动“调整大小”句柄手动调整列的大小，flex将自动对该列禁用。

下面的示例展示了flex的实际应用。注意事项如下:  
列A是固定大小。您可以用拖动手柄调整它的大小，其他两列将调整以填充可用空间  
列B有flex: 2, minWidth: 200和maxWidth: 350，所以它应该被限制在这个max/min宽度。  
列C有flex: 1，所以应该是列B大小的一半，除非列B受到minWidth/maxWidth规则的限制，在这种情况下，它应该占用剩余的可用空间。

{

headerName: 'Flexed Columns',

children: [

{

headerName: 'B',

minWidth: 200,

maxWidth: 350,

flex: 2,

},

{

headerName: 'C',

flex: 1,

},

在呈现数据时调整列的大小  
有两个主要的场景，你可能想要根据网格数据调整列的大小:  
1 行数据在网格初始化时可用  
2 行数据在网格初始化后可用，通常是在通过服务器调用异步设置数据之后

在第一种情况下，您可以在gridReady或firstDataRendered事件中触发autoSizeColumns()，因为在网格准备好时行数据已经呈现。

然而，在第二种情况下，您只能可靠地使用firstDataRendered，因为行数据将可用，从而呈现

组件

为了使ag-Grid能够使用您的Angular组件，您需要在顶层模块中提供它们：

@NgModule({

imports: [

AgGridModule.withComponents(

[

*SquareComponent*,

CubeComponent,

// ...other components

然后，您可以将这些组件用作编辑器，渲染器或过滤器。例如，要将Angular组件用作单元渲染器，您可以执行以下操作：

let colDefs = [

{

headerName: "Square Component",

field: "value",

cellRendererFramework: *SquareComponent*,

},

...other column definitions

]

子组件与父组件的沟通

有多种方法可以管理Angular中的组件通信（共享服务，局部变量等），但是您通常需要一种简单的方法来让“父”组件知道“子”组件上发生了某些事情。在这种情况下，最简单的方法是使用gridOptions.context持有对父项的引用，子项随后可以访问该引用。

// 在父组件中——该组件托管ag-grid-angular，并指定在网格中使用哪些angular组件

constructor() {

this.gridOptions = <GridOptions>{

context: {

componentParent: this

}

};

this.gridOptions.rowData = this.createRowData();

this.gridOptions.columnDefs = this.createColumnDefs();

}

// 在子组件中——在网格中动态创建的angular组件

// 父组件可以按如下方式访问:

this.params.context.componentParent

请注意，尽管我们在componentParent这里使用了属性名，但它可以是任何东西-重点是您可以使用该context机制在组件之间共享信息。

1 - 父组件

<https://github.com/ag-grid/ag-grid-angular-example/blob/master/systemjs_aot/app/dynamic-component-example/dynamic.component.ts>

2 - 子组件

<https://github.com/ag-grid/ag-grid-angular-example/blob/master/systemjs_aot/app/dynamic-component-example/child-message.component.ts>

筛选

网格可以使用过滤显示提供的行的子集。

1 列过滤器：列过滤器出现在列菜单和/或“过滤器工具面板”中。设置在列上的过滤器仅使用该列中的数据进行过滤。

2 快速过滤器：快速过滤器是给网格的一段文本（通常用户会在应用程序的某个位置键入它），用于使用网格中所有列中的数据过滤行。

3 外部过滤器：外部过滤器是应用程序独立于网格进行的任何过滤而过滤出行的一种机制。

最小高度与自动高度

显示行自动高度时，最小高度为50px。这是为了美观，特别是当网格中没有行时，允许显示“no rows”消息的空间，否则这个消息将覆盖在头部，这看起来不太好。  
当使用auto-height时，不可能指定最大高度。

Flexbox容器内  
默认情况下，网格运行一个计时器来监视其容器大小并相应地调整UI的大小。这可能会干扰display: flex set元素的默认行为。简单的解决方法是在网格元素的父元素中添加overflow: hidden。  
在新选项卡中打开下面的示例并调整窗口大小，以查看网格实例如何相应地调整大小。