一、

1 -

tyarn add @angular/flex-layout @angular/cdk

2 -

import { FlexLayoutModule } from '@angular/flex-layout';

@NgModule({

imports: [ FlexLayoutModule ],

});

配置好之后，你可以在HTML标签中使用Angular布局属性来实现flex布局:

<div fxLayout="row" fxLayoutAlign="space-between">

</div>

二、Angular Layout

Angular Layout使用Flexbox、CSS Grid和mediaQuery提供了一个复杂的API。这个模块使用自定义布局API、mediaQuery observables以及注入的DOM flexbox-2016和CSS网格样式为Angular开发者提供了组件布局特性。

布局引擎可以智能地自动化将适当的CSS应用到浏览器视图层次结构的过程。这种自动化还解决了传统的、手动的、只支持CSS的Flexbox CSS应用程序遇到的许多复杂问题和解决方案。

Angular Layout是一个纯粹的typescript布局引擎;不像其他Flexbox库中发布的纯CSS实现和AngularJS材质布局的JS+CSS实现。

Angular Layout 的实现是独立于Angular Material(v1或v2)

Angular Layout 实现目前只适用于Angular (v4.1及更高版本)应用程序

三、Angular Layout API

API是一个直观的HTML指令(又名属性)列表，可以用于HTML容器和元素。与使用传统的CSS样式表不同，开发人员将直接在HTML中声明式地定义其布局。

一个重要的基本概念是理解哪些api用于DOM容器，哪些api用于那些容器中的DOM子元素。

1 - 静态API概述

DOM容器的API

|  |  |  |
| --- | --- | --- |
| HTML | API | Allowed values |
| [fxLayout](https://github.com/angular/flex-layout/wiki/fxLayout-API" \o "https://github.com/angular/flex-layout/wiki/fxLayout-API) | <direction> [wrap] | row | column | row-reverse | column-reverse |
| [fxLayoutAlign](https://github.com/angular/flex-layout/wiki/fxLayoutAlign-API" \o "https://github.com/angular/flex-layout/wiki/fxLayoutAlign-API) | <main-axis> <cross-axis> | main-axis: start | center | end | space-around | space-between | space-evenly; cross-axis: start | center | end | stretch | space-between | space-around | baseline |
| [fxLayoutGap](https://github.com/angular/flex-layout/wiki/fxLayoutGap-API" \o "https://github.com/angular/flex-layout/wiki/fxLayoutGap-API) | % | px | vw | vh |  |
| [gdAlignColumns](https://github.com/angular/flex-layout/wiki/gdAlignColumns-API" \o "https://github.com/angular/flex-layout/wiki/gdAlignColumns-API) | <main-axis> <cross-axis> | main-axis: start | center | end | space-around | space-between | space-evenly | stretch; cross-axis: start | center | end | stretch |
| [gdAlignRows](https://github.com/angular/flex-layout/wiki/gdAlignRows-API" \o "https://github.com/angular/flex-layout/wiki/gdAlignRows-API) | <main-axis> <cross-axis> | main-axis: start | center | end | space-around | space-between | space-evenly | stretch; cross-axis: start | center | end | stretch |
| [gdAreas](https://github.com/angular/flex-layout/wiki/gdAreas-API" \o "https://github.com/angular/flex-layout/wiki/gdAreas-API) | names separated by |, e.g. gdAreas="area1 | area2 | area3" | gdInline for inline-grid |
| [gdAuto](https://github.com/angular/flex-layout/wiki/gdAuto-API" \o "https://github.com/angular/flex-layout/wiki/gdAuto-API) | row | column | dense | row dense | column dense | gdInline for inline-grid |
| [gdColumns](https://github.com/angular/flex-layout/wiki/gdColumns-API" \o "https://github.com/angular/flex-layout/wiki/gdColumns-API) | any valid input for grid-template-columns | gdInline for inline-grid |
| ! at the end means grid-auto-columns |
| [gdRows](https://github.com/angular/flex-layout/wiki/gdRows-API" \o "https://github.com/angular/flex-layout/wiki/gdRows-API) | any valid input grid-template-rows | gdInline for inline-grid |
| ! at the end means grid-auto-rows |
| [gdGap](https://github.com/angular/flex-layout/wiki/gdGap-API" \o "https://github.com/angular/flex-layout/wiki/gdGap-API) | % | px | vw | vh | gdInline for inline-grid |

这些指令会影响容器中的流和布局子元素

DOM元素的API

|  |  |
| --- | --- |
| HTML | Allowed values |
| [fxFlex](https://github.com/angular/flex-layout/wiki/fxFlex-API" \o "https://github.com/angular/flex-layout/wiki/fxFlex-API) | "" | px | % | vw | vh | <grow> <shrink> <basis>, |
| [fxFlexOrder](https://github.com/angular/flex-layout/wiki/fxFlexOrder-API" \o "https://github.com/angular/flex-layout/wiki/fxFlexOrder-API) | int |
| [fxFlexOffset](https://github.com/angular/flex-layout/wiki/fxFlexOffset-API" \o "https://github.com/angular/flex-layout/wiki/fxFlexOffset-API) | % | px | vw | vh |
| [fxFlexAlign](https://github.com/angular/flex-layout/wiki/fxFlexAlign-API" \o "https://github.com/angular/flex-layout/wiki/fxFlexAlign-API) | start | baseline | center | end |
| [fxFlexFill, fxFill](https://github.com/angular/flex-layout/wiki/fxFlexFill-API" \o "https://github.com/angular/flex-layout/wiki/fxFlexFill-API) |  |
| [gdArea](https://github.com/angular/flex-layout/wiki/gdArea-API" \o "https://github.com/angular/flex-layout/wiki/gdArea-API) | string name for the area as defined in gdAreas |
| [gdColumn](https://github.com/angular/flex-layout/wiki/gdColumn-API" \o "https://github.com/angular/flex-layout/wiki/gdColumn-API) | any valid value for grid-column |
| [gdRow](https://github.com/angular/flex-layout/wiki/gdRow-API" \o "https://github.com/angular/flex-layout/wiki/gdRow-API) | any valid value for grid-row |
| [gdGridAlign](https://github.com/angular/flex-layout/wiki/gdGridAlign-API" \o "https://github.com/angular/flex-layout/wiki/gdGridAlign-API) | <row-axis> <cross-axis> |
| [gdRows](https://github.com/angular/flex-layout/wiki/gdRows-API" \o "https://github.com/angular/flex-layout/wiki/gdRows-API) | any valid input grid-template-rows |
|
| [gdGap](https://github.com/angular/flex-layout/wiki/gdGap-API" \o "https://github.com/angular/flex-layout/wiki/gdGap-API) | % | px | vw | vh |
|  |  |

这些指令会影响主机元素的布局和大小。注意，API期望它们的宿主元素在DOM容器中(一个“block”元素，它本身使用容器的布局API)。

任何元素的API:

|  |  |
| --- | --- |
| HTML API | Allowed values |
| [fxHide](https://github.com/angular/flex-layout/wiki/fxHide-API" \o "https://github.com/angular/flex-layout/wiki/fxHide-API) | TRUE | FALSE | 0 | "" |
| [fxShow](https://github.com/angular/flex-layout/wiki/fxShow-API" \o "https://github.com/angular/flex-layout/wiki/fxShow-API) | TRUE | FALSE | 0 | "" |
| [ngClass](https://github.com/angular/flex-layout/wiki/ngClass-API" \o "https://github.com/angular/flex-layout/wiki/ngClass-API) | [@extends ngClass core](https://angular.io/api/common/NgClass" \o "https://angular.io/api/common/NgClass) |
| [ngStyle](https://github.com/angular/flex-layout/wiki/ngStyle-API" \o "https://github.com/angular/flex-layout/wiki/ngStyle-API) | [@extends ngStyle core](https://angular.io/api/common/NgStyle" \o "https://angular.io/api/common/NgStyle) |
| [imgSrc](https://github.com/angular/flex-layout/wiki/imgSrc-API" \o "https://github.com/angular/flex-layout/wiki/imgSrc-API) | @extends src attribute |

下面是使用容器和元素静态API的HTML标记示例:

<div fxLayout='column' class="zero">

<div fxFlex="33" class="one" ></div>

<div fxFlex="33%" [fxLayout]="direction" class="two">

<div fxFlex="22%" class="two\_one"></div>

<div fxFlex="205px" class="two\_two"></div>

<div fxFlex="30" class="two\_three"></div>

</div>

<div fxFlex class="three"></div>

</div>

Angular布局指令将CSS样式直接内联分配给宿主元素。这些内联样式覆盖了元素:host上的继承样式、阴影dom样式甚至阴影dom树样式

Angular Layout还有一组巨大的响应特性，可以让开发者轻松地针对不同的显示设备更改UX布局配置。请参阅响应API页面上的文档。

2 - 响应API

响应式布局的Material Design适应任何可能的屏幕大小。谷歌的规范提供了指导，包括确保布局一致性的灵活网格，关于内容如何在不同屏幕上回流的断点细节，以及对应用程序如何从小屏幕扩展到超大屏幕的描述。

<https://github.com/angular/flex-layout/wiki/Responsive-API>

开发人员应该参考HTML声明式API以获得特定的静态API细节，然后通过添加响应后缀(如下所述)简单地扩展HTML标记的用法!  
Angular Layout会自动处理所有关于监听mediaQuery激活并将响应值应用到主机DOM元素的细节。

（1 ）响应特性  
为了使用响应特性扩展静态API，我们将首先将特定的断点别名与mediaQuery值关联起来。

1. MediaQueries和别名  
   我们可以使用断点别名将断点与mediaQuery定义关联起来:

|  |  |
| --- | --- |
| breakpoint | mediaQuery |
| xs | 'screen and (max-width: 599px)' |
| sm | 'screen and (min-width: 600px) and (max-width: 959px)' |
| md | 'screen and (min-width: 960px) and (max-width: 1279px)' |
| lg | 'screen and (min-width: 1280px) and (max-width: 1919px)' |
| xl | 'screen and (min-width: 1920px) and (max-width: 5000px)' |
|  |  |
| lt-sm | 'screen and (max-width: 599px)' |
| lt-md | 'screen and (max-width: 959px)' |
| lt-lg | 'screen and (max-width: 1279px)' |
| lt-xl | 'screen and (max-width: 1919px)' |
|  |  |
| gt-xs | 'screen and (min-width: 600px)' |
| gt-sm | 'screen and (min-width: 960px)' |
| gt-md | 'screen and (min-width: 1280px)' |
| gt-lg | 'screen and (min-width: 1920px)' |

如果我们结合断点别名与静态API，我们可以轻松地支持响应式断点使用一个简单的标记约定:

别名用作后缀扩展到静态API HTML标记:

<api>.<breakpoint alias>="<value>"

[<api>.<breakpoint alias>]="<expression>"

下面是一个使用响应式布局API的例子:

<div fxLayout='column' class="zero">

<div fxFlex="33" [fxFlex.md]="box1Width" class="one" ></div>

<div fxFlex="33" [fxLayout]="direction" fxLayout.md="row" class="two">

<div fxFlex="22" fxFlex.md="10px" fxHide.lg class="two\_one"></div>

<div fxFlex="205px" fxFlex.md="65" class="two\_two"></div>

<div fxFlex="30px" fxFlex.md="25" fxShow[fxHide.md]="hideBox" lass="two\_three">

</div>

</div>

<div fxFlex class="three"></div>

</div>

在HTML API上面的标记中，指令同时使用静态值和表达式绑定;其中值以原始值、百分比值或像素值表示。

注意:数值没有显式标注为px、vw或vh，默认为百分比值。

断点激活回退算法

当激活一个断点，并且宿主元素没有为新激活的断点定义响应API时，Flex-Layout响应引擎使用回退、下降扫描算法来确定替换激活值。该算法使用断点优先级按向下优先级对激活断点进行排序。

该算法搜索:  
对于非重叠断点:搜索从最小到最大的断点范围扫描，以找到最近的、匹配的激活值。

(**xs**, **sm**, **md**, **lg**, **xl**)

对于重叠断点:搜索从最小到最大的断点范围扫描，以找到最近的、匹配的激活值。

* (**lt-sm**, **lt-md**, **lt-lg**, **lt-xl**);lt-xl的范围最大
* (**gt-xs**, **gt-sm**, **gt-md**, **gt-lg**); gt-xs 是 最大 的 range.

<div fxShow fxHide.xs="false" fxHide.lg="true"> ... </div>

激活的断点为:

xl, 然后退回到默认的fxShow;这样就显示了div

lg, 然后隐藏div(因为value === 'true')

md, 然后返回到默认的fxShow;这样就显示了div

sm, 然后返回到默认的fxShow;这样就显示了div

xs, 然后显示div(因为value === 'false')

<div fxFlex="50%" fxFlex.gt-sm="100%"> ... </div>

When the activated breakpoint is:

xl, then fallback to 'gt-sm' so the div sizing is 100%

lg, then fallback to 'gt-sm' so the div sizing is 100%

md, then fallback to 'gt-sm' so the div sizing is 100%

sm, then fallback to the default fxFlex="50%"; so the div sizing is 50%

xs, then fallback to the default fxFlex="50%"; so the div sizing is 50%

3 - 编程接口文档

大多数Angular布局功能是通过模板HTML中声明式使用的指令提供的。但是，有两个(2)编程式功能，被发布用于编程式用途:

（1）MediaObserver:  
用于订阅MediaQuery激活更改的可注入服务。您可以通过media$属性访问其可观察性

import {MediaObserver} from '@angular/flex-layout';

constructor(public mediaObserver: MediaObserver ) {

mediaObserver.media$.subscribe(...);

}

（2）断点:  
注入令牌，用于使用自定义MediaQuery断点覆盖或扩展默认断点。

import {BREAKPOINT} from '@angular/flex-layout';

providers: [{provide: BREAKPOINT, useValue: MY\_CUSTOM\_BREAKPOINTS, multi: true}]

HTML API(声明)

Angular布局的特性最好以声明的方式用在模板HTML中。这个API有两个重要的特性集:  
静态API:静态API特性的总结。  
响应API:引入响应API和断点细节。

由于库中的每个指令(又名API)都有自己的约束和选项，下面的链接应该用于导航到每个指令的特定文档页面

容器  
这个API集对有1个或多个嵌套子元素的DOM容器元素应用CSS样式:

fxLayout: 定义flexbox容器内子项目的流顺序

<div fxLayout="row" fxLayout.xs="column"></div>

fxLayoutGap: 定义flexbox容器中的子项是否有间隔

<div fxLayoutGap="10px"></div>

fxLayoutAlign: 定义flexbox容器中flexbox项如何根据主轴和交叉轴对齐

<div fxLayoutAlign="start stretch"></div>

gdAlignColumns: 按列排列条目 column-wise

<div gdAlignColumns="start stretch"></div>

gdAlignRows:将项目行对齐row-wise

<div gdAlignRows="start stretch"></div>

gdAreas: 描述网格包含的区域

<div gdAreas="area1 | area2 | area3"></div>

gdAuto: 控制网格的自动位置

<div gdAuto="row dense"></div>

gdColumns: 定义网格列的行名称和跟踪大小调整函数

<div gdColumns="60px 60px"></div>

gdRows: 定义网格行的行名和跟踪大小调整函数

<div gdRows="60px 60px"></div>

gdGap: 定义网格项之间的间隙

<div gdGap="60px 60px"></div>

容器中的子元素  
这个API集为嵌套在DOM容器中的DOM元素应用CSS样式:

fxFlex: 此标记指定flexbox容器流中其宿主元素的大小

<div fxFlex="1 2 calc(15em + 20px)"></div>

fxFlexOrder: 定义flexbox项的顺序

<div fxFlexOrder="2"></div>

fxFlexOffset: 在父容器流布局中偏移flexbox项

<div fxFlexOffset="20px"></div>

fxFlexAlign: 类似于fxLayoutAlign，但只适用于单个flexbox项目，而不是所有项目

<div fxFlexAlign="center"></div>

fxFlexFill:使布局容器中元素的宽度和高度最大化

<div fxFlexFill></div>

gdArea: 将元素标记为CSS网格区域

<div gdArea="area1"></div>

gdColumn: 指定网格项的大小和在网格列中的位置

<div gdColumn="1 / span 2"></div>

gdRow: 指定网格项的大小和在网格行的位置

<div gdRow="1 / span 2"></div>

gdGridAlign: 对齐容器元素中的网格项

<div gdGridAlign="start stretch"></div>

特殊的响应特性

虽然以下api也添加或删除DOM元素的内联样式，但它们不是FlexBox CSS特有的。  
相反，这些响应api用于调整特定的，非flexbox样式时，特定的媒体查询已激活:

fxShow: 此标记指定是否应该显示(或不显示)其主机元素

<div fxShow [fxShow.xs]="isVisibleOnMobile()"></div>

fxHide: 此标记指定是否不应显示其主机元素

<div fxHide [fxHide.gt-sm]="isVisibleOnDesktop()"></div>

ngClass : 使用基于mediaQuery激活的类更改增强了ngClass指令

<div [ngClass.sm]="{'fxClass-sm': hasStyle}"></div>

ngStyle: 使用基于mediaQuery激活的样式更新增强了ngStyle指令

<div [ngStyle.xs]="{'font-size.px': 10, color: 'blue'}"></div>