# Echarts在Angular12中的使用

ECharts可以提供直观，交互丰富，可高度个性化定制数据可视化图表，是开发者手上不可多得的一把利器。比较详细的引导教程对新手玩家也比较友好，但美中不足的是仅有一些新手村教程，对于angular中使用echarts教程官网上没有提及到，本文从基本使用、父组件获取数据传给子组件进行渲染图表、使用liquidfill插件实现水球图、通过指令使用echarts这四种场景进行剖析echarts在angular12中的使用。

# Echarts基本使用

这一小节讲的是echarts怎么引入我们的angular项目并使用。官方文档对于echarts的引入讲述的比较清楚，这里主要讲解一下它的使用，更多详情可以点击官网查看：

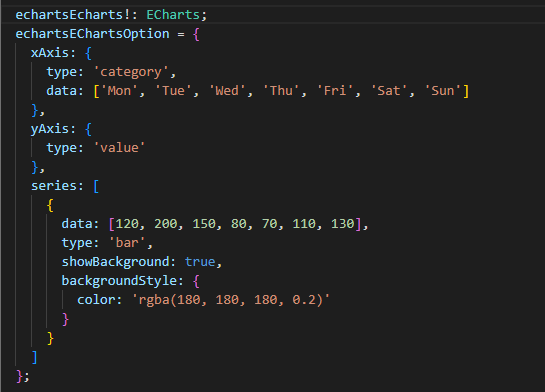
<https://echarts.apache.org/handbook/zh/get-started/>

## 项目引入：NPM 安装 ECharts

npm install echarts --save

## 定义Echarts和Option

下面提供的两种方法都需要定义Echarts对象和Option。我们在ts文件中先进行定义。



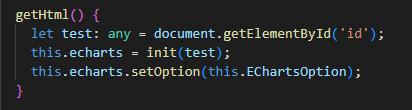
## 1.使用id获取DOM元素

JS的原生方法获取dom，对于不熟悉angular的伙伴来说使用简单，但不建议使用。

Html文件代码

<div id="id"></div>

Ts文件，通过函数获取DOM，初始化图表设置数据



## 使用ViewChild获取DOM元素

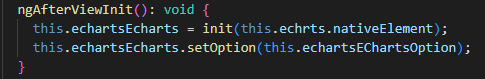
上面的方法是通过js原生方法获取元素属性的，angular中的@ViewChild可以理解未一个指代，我们通过这个指代可以得到组件或者元素，并可以使用这个组件的值和方法。推荐使用下面@ViewChild方法。

<div #echrts id="echrtsView"></div>

获取DOM元素

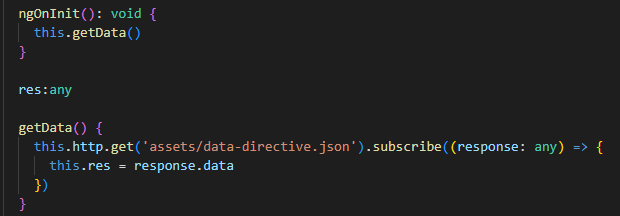
@ViewChild('echrts') echrts: any;

初始化设置图表数据



# 父组件传值给子组件

## 创建子组件后，父组件获取数据



## 通过input传给子组件

其中父组件html这么写：



子组件ts接收：

@Input() res: any;

## 子组件通过ViewChild获取元素绘制图表

1. 引入并使用ViewChild

Ts文件使用

@ViewChild("echart") echartdom: any;

Html标记

<div #echart class="echart">

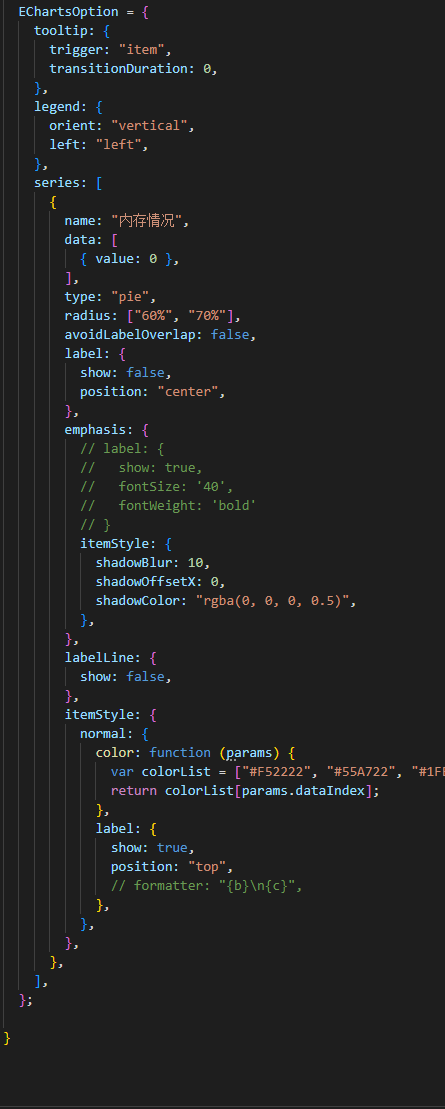
</div>

1. 子组件引入并使用Echarts

import { ECharts, init } from "echarts";

echart!: ECharts;

1. 建立option



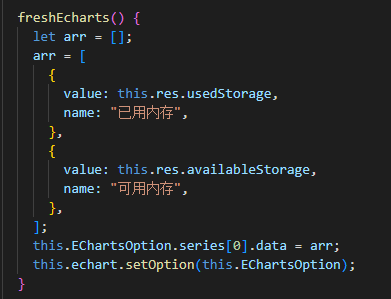
1. 在子组件初始化echarts

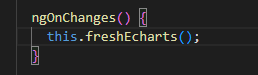


初始化的过程在视图即将出现的生命周期函数ngAfterViewInit中

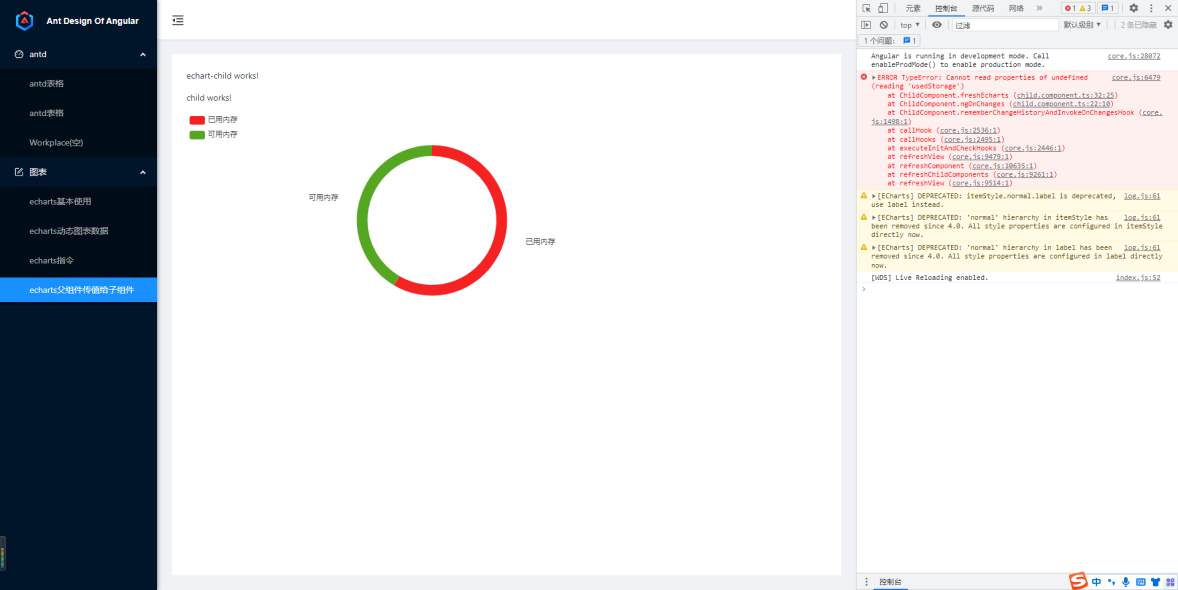
1. 数据驱动更新

下面这是更新echarts图表的函数，在ngOnChanges生命周期函数中调用。当有数据改变时就会触发更新。





1. 此时页面显示出图表，但是页面有报错

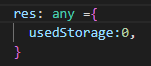


## 解决报错

（1）属性不存在

core.js:6479 ERROR TypeError: Cannot read properties of undefined (reading 'usedStorage')

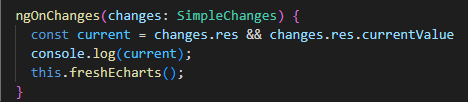
我们前面是把整个返回的数据传给子组件的，这里说读取不到这个属性，那么我们可以在父组件进行定义一个空值。代码如下：



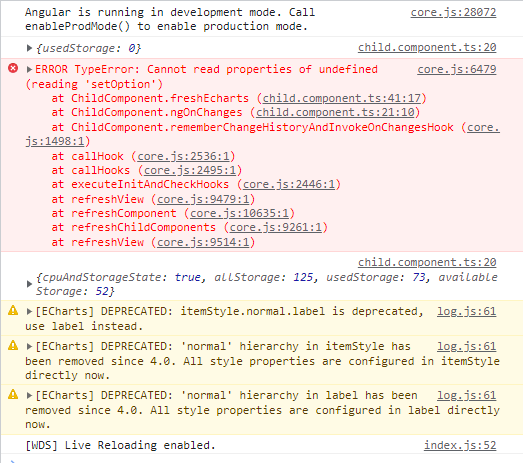
（2）无法设置图表配置项setOption

core.js:6479 ERROR TypeError: Cannot read properties of undefined (reading 'setOption')

这里就是子组件ngOnChanges使用问题了。我们打印看看这里使用的具体信息。



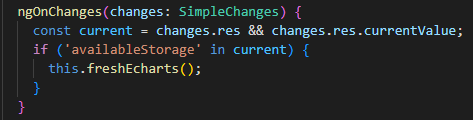
这里打印changes表现不是特别直观，我们就打印current的值。



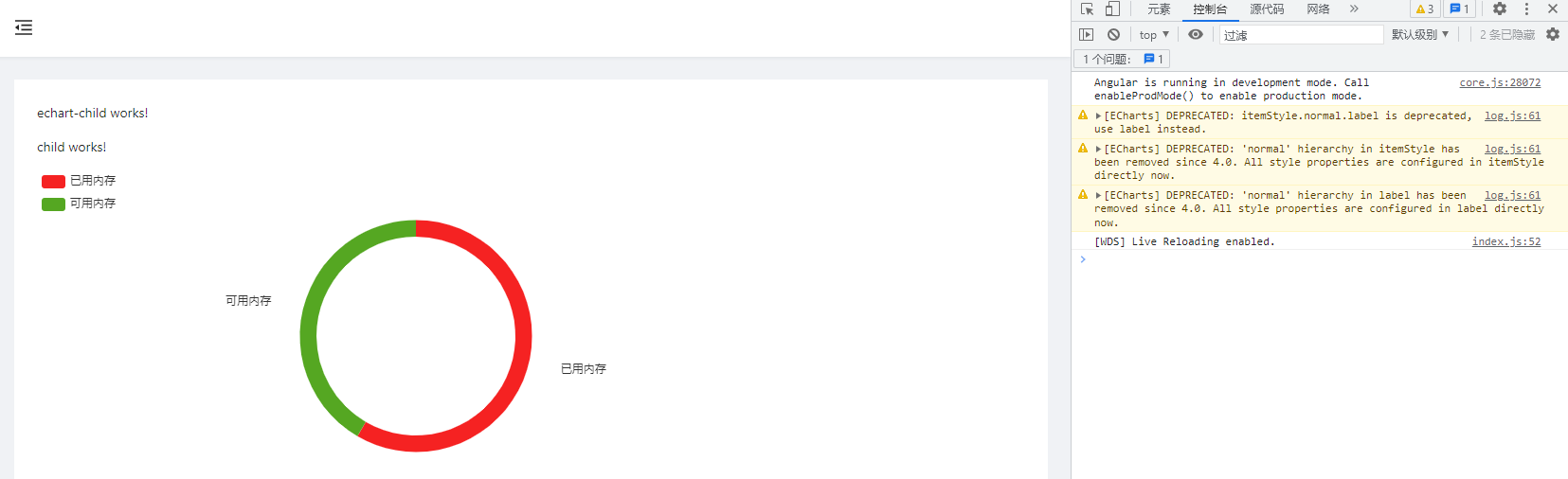
由打印结果可以得知，第一次打印输出为未get获取到数据的父组件初始值，第二次打印输出为正常获取到的数据，但是在第一次未获取正常数据之前，已经进行了一次更新图表，这就是报错的原因。解决方案很简单，让更新图表的函数选择性执行就行。编码思路：

我们刚刚打印得知，前后相差几个变量，那么我们判断这相差的变量是否存在即可。

代码如下：



这样就可以正常显示而且没有报错了。



# Echarts使用水球图

水球图属于填充仪表盘类，其生动的动画效果可以很好展示百分比数据。

## 下载插件

yarn add echarts-liquidfill

## 在main.ts中注册

import 'echarts-liquidfill';

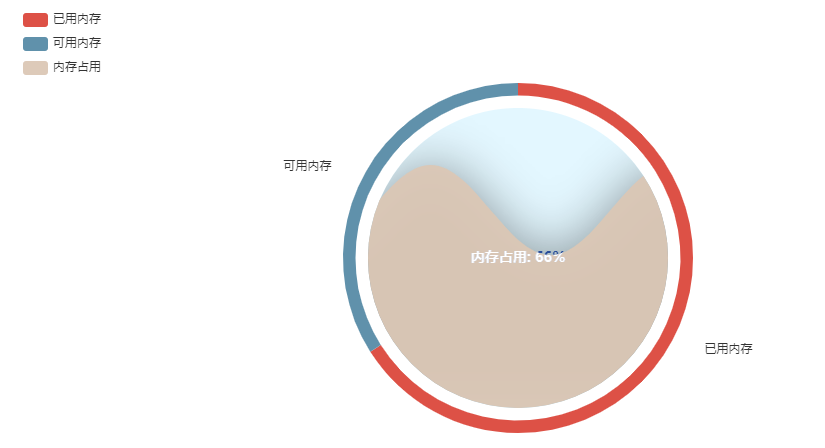
## 进行使用

使用方式和echarts没有太大区别，主要是在option中修改参数。

Series数组再加入一个对象：



## 效果图如下



## 参数讲解

### 类型type: 'liquidFill',

这是水球图实现的关键，type声明了这个的图的属性。

官方讲解中这里的type有这些属性：

line（折线图）、bar（柱状图）、pie（饼图）、scatter（散点图）、effectScatter（气泡图）、radar（雷达图）、tree（树图）、treemap（树状数据图）、sunburst（旭日图）、boxplot（箱形图）、candlestick（K线图）、heatmap（热力图）、map（地图）、parallel（平行坐标系）、lines（路径图）、graph（关系图）、sankey（桑基图）、funnel（漏斗图）、gauge（仪表盘）、pictorialBar（象形柱图）、themeRiver（主题河流）、custom（自定义系列）。

### 多重波浪

通过设置data的多个值可以展现多个波浪。

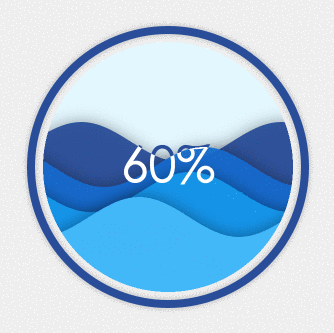
var option = {

series: [{

type: 'liquidFill',

data: [0.6, 0.5, 0.4, 0.3]

}]};

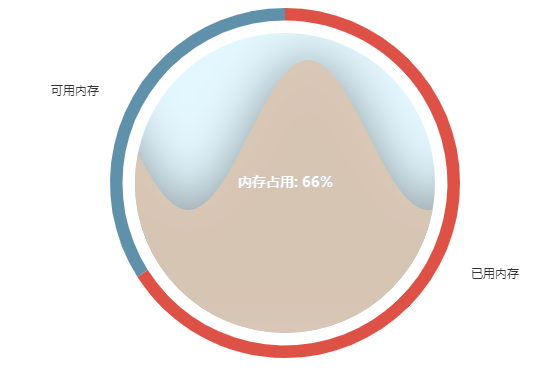


### 振幅amplitude

波浪的曲折程度可以理解为水波图的振幅，当振幅为0时水波纹就是平的。



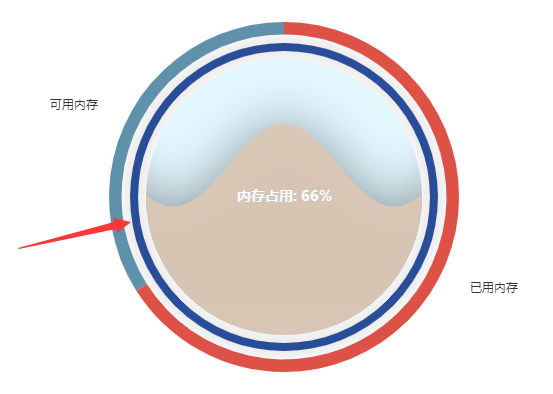
设置值大的时候展现就和山峰一样。



### 内环圈设置outline

outline: {show: false}

这里的show值显示TRUE时会显示这里箭头处的内环圈。为FALSE时则会隐藏。



### **形状Shape**

这个形状可以根据需求进行修改，比如方形、圆形、三角形等。

shape: 'diamond'



参数较多，感兴趣的伙伴可以阅读github讲解。

<https://github.com/ecomfe/echarts-liquidfill>

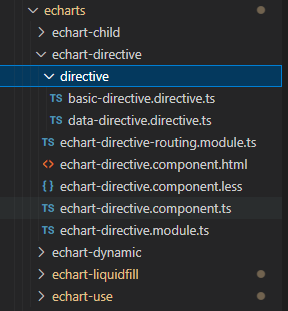
# Echarts指令

## 在目录下创建指令

ng g directive dataDirective

## 在module引入指令

创建指令时可能在module文件中进行引入，如果没有引入就需要手动引入。



我们在echart-directive文件下单独建立directive文件夹，里面创建指令文件。如果没有自动引入，这里就需要在echart-directive.module.ts文件中手动进行引入。



## Html文件中使用指令

引入比较简单，在创建指令文件的时候，指令文件有这么几行代码，里面写了这个指令的名称appBasicDirective。

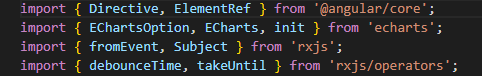


我们之间在div中进行引入就行。

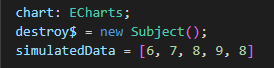
<div appBasicDirective class="echart"></div>

## 在指令ts文件中进行编写代码

### 引入依赖



### 定义echarts对象和其他内容



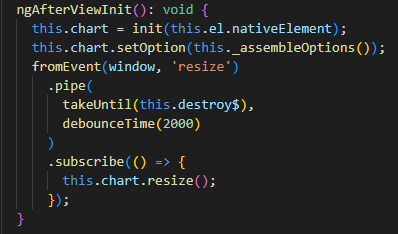
### 构造器注册



### 定义Option



### 初始化并设置数据



TakeUntil 函数官方文档对这个操作符的解释是：

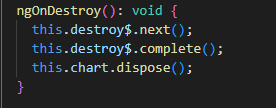
Emit values until provided observable emits.

即它可以被赋予另一个起锚定作用的 Observable，当该锚定 Observable emit 值时，原始的 Observable 就停止发射值，进入 complete 操作。或者说到达一个条件后取消订阅。

debounceTime函数用于防抖。

resize 函数的作用是改变图表尺寸，在容器大小发生改变时需要手动调用。

1. 销毁图表实例



dispose函数的作用是销毁实例，实例销毁后无法再被使用。

# 结语

以上四种场景对echarts的使用就介绍到这了！他们可以划分为两类，一种就是直接写在component.ts里面，另外一种就是把echarts抽离出来写成指令，个人比较推荐写成指令抽离出来，这样可以让component.ts文件更整洁，尤其是业务逻辑多的情况下对echarts的抽离显得格外重要！