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。

```
echartsEcharts!: ECharts;
echartsEChartsOption = {
  xAxis: {
    type: 'category',
    data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
  },
  yAxis: {
    type: 'value'
  },
  series: [
    {
      data: [120, 200, 150, 80, 70, 110, 130],
      type: 'bar',
      showBackground: true,
      backgroundStyle: {
        color: 'rgba(180, 180, 180, 0.2)'
      }
    }
  ]
};
```

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

```
getHtml() {
  let test: any = document.getElementById('id');
  this.echarts = init(test);
  this.echarts.setOption(this.EChartsOption);
}
```

2. 使用 ViewChild 获取 DOM 元素

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

获取 DOM 元素

```
@ViewChild('echrts') echrts: any;
```

初始化设置图表数据

```
ngAfterViewInit(): void {
  this.echartsEcharts = init(this.echrts.nativeElement);
  this.echartsEcharts.setOption(this.echartsEChartsOption);
}
```

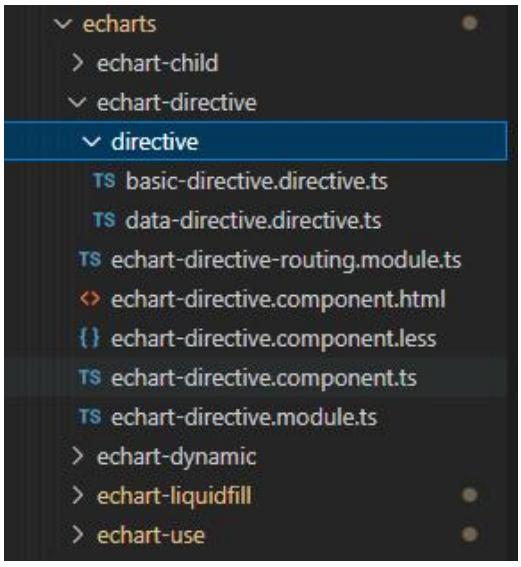
Echarts 指令

1. 在目录下创建指令

ng g directive dataDirective

2. 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。

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { EchartDirectiveRoutingModule } from './echart-directive-routing.module';
import { EchartDirectiveComponent } from './echart-directive.component';
import { DataDirectiveDirective } from './directive/data-directive.directive';
import { BasicDirectiveDirective } from './directive/basic-directive.directive';

@NgModule({
  declarations: [
    EchartDirectiveComponent,
    DataDirectiveDirective,
    BasicDirectiveDirective,
  ],
  imports: [
    CommonModule,
    EchartDirectiveRoutingModule,
  ]
})

export class EchartDirectiveModule { }
```

3. 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。

```
@Directive({
  selector: '[appBasicDirective]'
})
```

我们之间在 div 中进行引入就行。

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

4. 在指令 ts 文件中进行编写代码

（1）引入依赖

```
import { Directive, ElementRef } from '@angular/core';
import { EChartsOption, ECharts, init } from 'echarts';
import { fromEvent, Subject } from 'rxjs';
import { debounceTime, takeUntil } from 'rxjs/operators';
```

（2）定义 echarts 对象和其他内容

```
chart: ECharts;
destroy$ = new Subject();
simulatedData = [6, 7, 8, 9, 8]
```

（3）构造器注册

```
constructor(
  public el: ElementRef
) { }
```

（4）定义 Option

```
private _assembleOptions(): EChartsOption {
  return {
    title: {
      text: '指令的基本使用'
    },
    xAxis: {
      type: 'category',
      data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    },
    yAxis: {
      type: 'value'
    },
    series: [
      {
        data: [150, 230, 224, 218, 135, 147, 260],
        type: 'line'
      }
    ]
  };
}
```

（5）初始化并设置数据

```
ngAfterViewInit(): void {
  this.chart = init(this.el.nativeElement);
  this.chart.setOption(this._assembleOptions());
  fromEvent(window, 'resize')
    .pipe(
      takeUntil(this.destroy$),
      debounceTime(2000)
    )
    .subscribe(() => {
      this.chart.resize();
    });
}
```

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。

（6）销毁图表实例

```
ngOnDestroy(): void {
  this.destroy$.next();
  this.destroy$.complete();
  this.chart.dispose();
}
```

dispose 函数的作用是销毁实例，实例销毁后无法再被使用。

Echarts 使用小结

通过调用 js 原生方法借助 id 获取元素属性的途径相对比较容易，不需要了解太多 angular 的知识，但也正因如此不能将框架的优势发挥出来。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。在组件内写表格的话比较推荐使用@ViewChild 方法。而指令这种方法可以将图表这一块逻辑单独抽离出来，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！

父组件传值给子组件

讲述完 echarts 在 angular 的基本使用后，我们看看两种应用场景，一种是本章节介绍的父组件传值给子组件，子组件获取到数据后通过 echarts 进行展示，另外一种是让页面比较炫酷的水球图。

1. 创建子组件后，父组件获取数据

```
ngOnInit(): void {
  this.getData()
}

res:any

getData() {
  this.http.get('assets/data-directive.json').subscribe((response: any) => {
    this.res = response.data
  })
}
```

2. 通过 input 传给子组件

父组件 html 代码：

```
<app-child [res]="res"></app-child>
```

子组件 ts 接收：

```
@Input() res: any;
```

3. 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>
```

（2）子组件引入并使用 Echarts

```
import { ECharts, init } from "echarts";
echart!: ECharts;
```

（3）建立 option

```
EChartsOption = {
  tooltip: {
    trigger: "item",
    transitionDuration: 0,
  },
  legend: {
    orient: "vertical",
    left: "left",
  },
  series: [
    {
      name: "内存情况",
      data: [
        { value: 0 },
      ],
      type: "pie",
      radius: ["60%", "70%"],
      avoidLabelOverlap: false,
      label: {
        show: false,
        position: "center",
      },
      emphasis: {
        // label: {
        //   show: true,
        //   fontSize: '40',
        //   fontWeight: 'bold'
        // }
        itemStyle: {
          shadowBlur: 10,
          shadowOffsetX: 0,
          shadowColor: "rgba(0, 0, 0, 0.5)",
        },
      },
      labelLine: {
        show: false,
      },
      itemStyle: {
        normal: {
          color: function (params) {
            var colorList = ["#F52222", "#55A722", "#1F
            return colorList[params.dataIndex];
          },
          label: {
            show: true,
            position: "top",
            // formatter: "{b}\n{c}",
          },
        },
      },
    },
  ],
};
```

（4）在子组件初始化 echarts

```
ngAfterViewInit(): void {
  this.echart = init(this.echartdom.nativeElement);
}
```

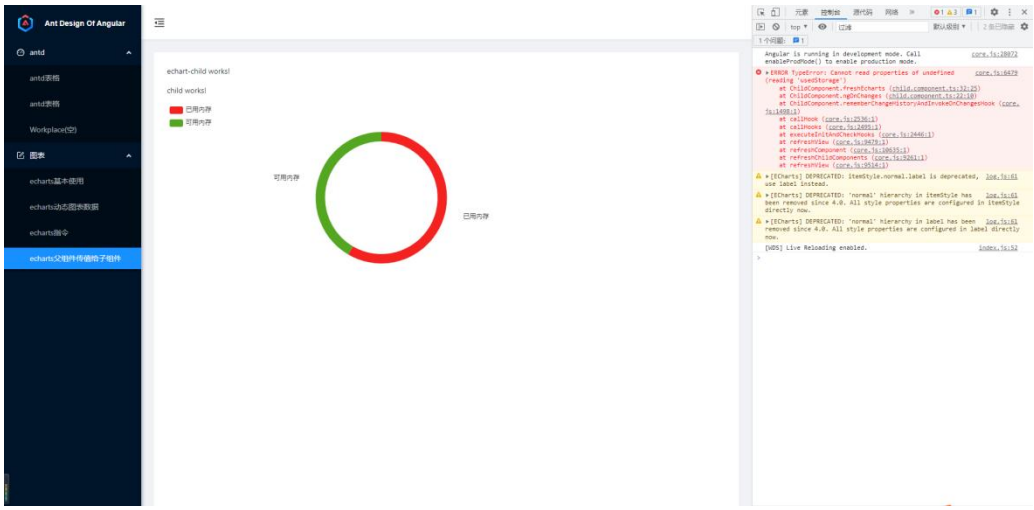
初始化的过程在视图即将出现的生命周期函数 ngAfterViewInit 中

（5）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。

```
freshEcharts() {
  let arr = [];
  arr = [
    {
      value: this.res.usedStorage,
      name: "已用内存",
    },
    {
      value: this.res.availableStorage,
      name: "可用内存",
    },
  ];
  this.EChartsOption.series[0].data = arr;
  this.echart.setOption(this.EChartsOption);
}
```

（6）此时页面显示出图表，但是页面有报错



4. 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：

```
res: any ={
  usedStorage:0,
}
```

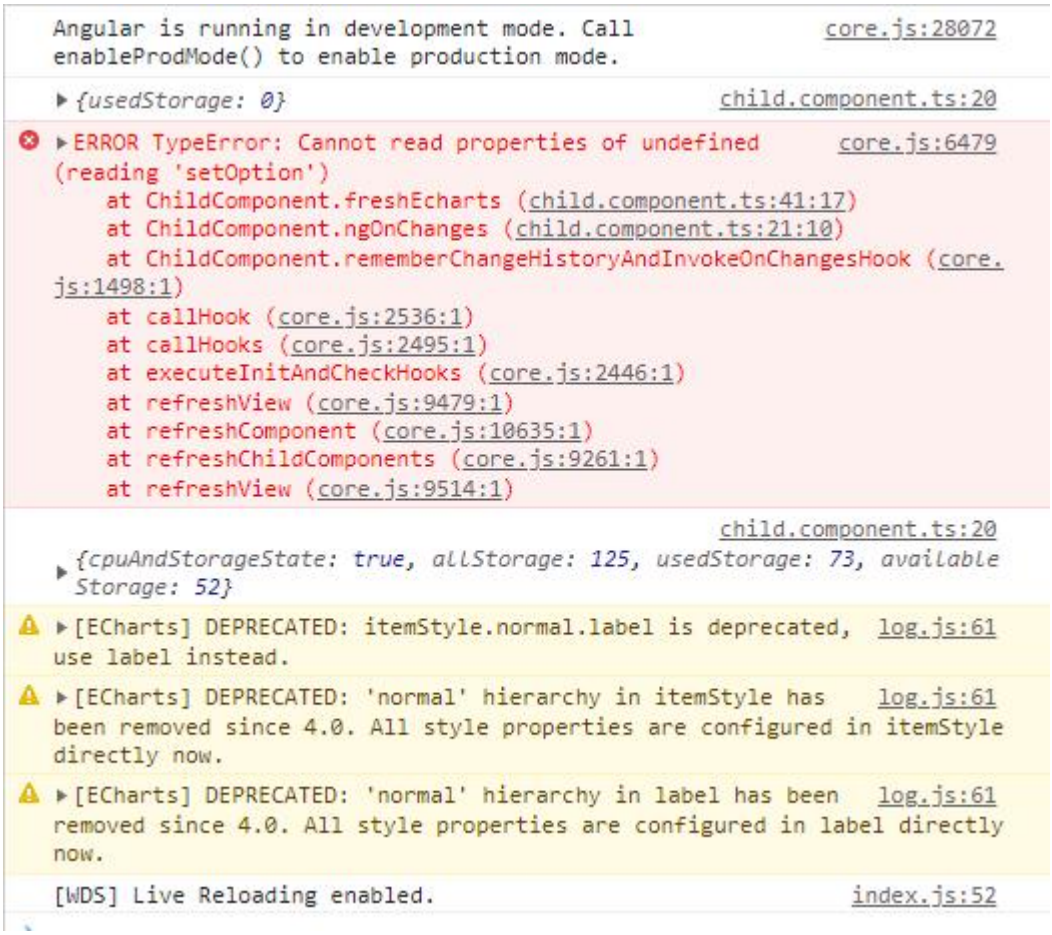
（2）无法设置图表配置项 setOption

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。

```
ngOnChanges(changes: SimpleChanges) {
  const current = changes.res && changes.res.currentValue
  console.log(current);
  this.freshEcharts();
}
```

这里打印 changes 表现不是特别直观，我们就打印 current 的值。



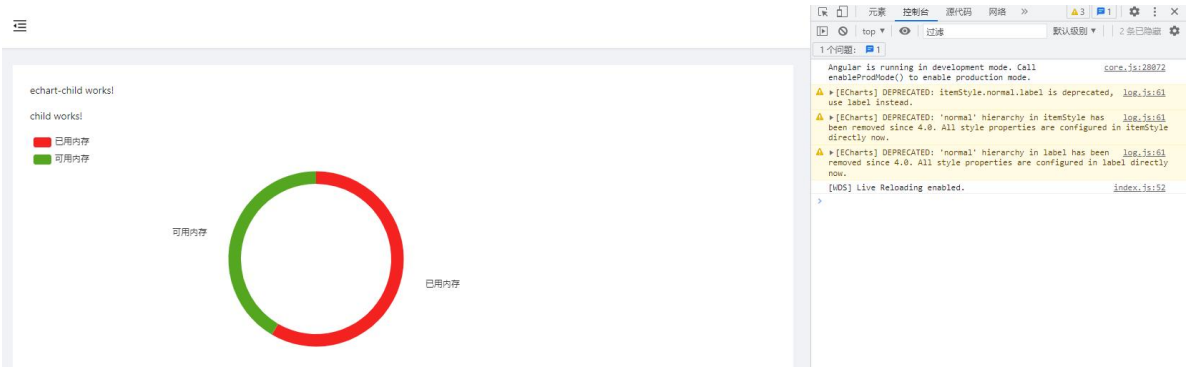
由打印结果可以得知，第一次打印输出为未 get 获取到数据的父组件初始值，第二次打印输出为正常获取到的数据，但是在第一次未获取正常数据之前，已经进行了一次更新图表，这就是报错的原因。解决方案很简单，让更新图表的函数选择性执行就行。编码思路：

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：

```
ngOnChanges(changes: SimpleChanges) {
  const current = changes.res && changes.res.currentValue;
  if ('availableStorage' in current) {
    this.freshEcharts();
  }
}
```

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。

1. 下载插件

yarn add echarts-liquidfill

2. 在 main.ts 中注册

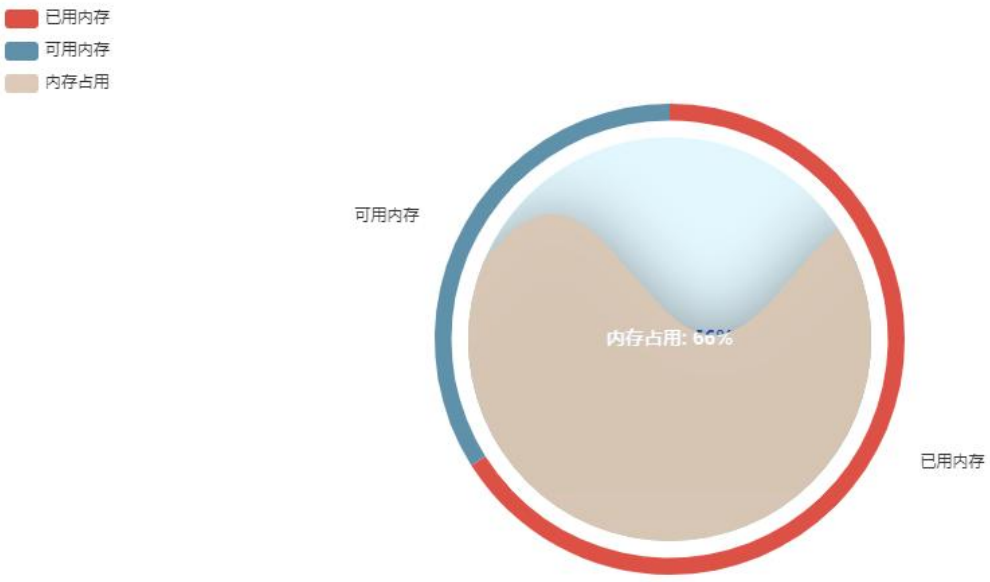
```
import 'echarts-liquidfill';
```

3. 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：

```
{
  name: "内存占用",
  // 内环圈
  outline: {
    show: false,
  },
  // 背景样式设置
  // backgroundStyle: {
  //   borderWidth: 5,
  //   borderColor: 'red',
  //   color: 'yellow'
  // },
  radius: "60%",
  type: "liquidFill",
  data: [
    2,
    {
      value: 0.6,
      direction: 'left', //波浪方向
    },
  ],
  amplitude: '15 %', //波浪的振幅
  color: ["#dbc7b5"],
  label: {
    formatter: function (param) {
      return `${param.seriesName}: ${param.value * 100}%`;
    },
    fontSize: 14,
  },
  tooltip: {
    formatter(param) {
      return `${param.seriesName}: ${param.value * 100}%`;
    },
  },
},
```

4. 效果图如下



5. 部分参数讲解

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。

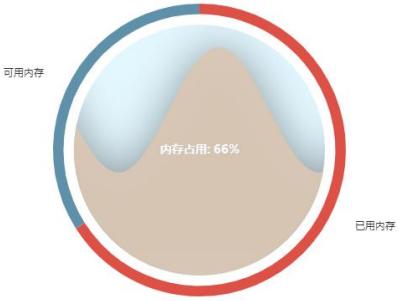
官方讲解中这里的 type 有这些属性：

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）。

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。

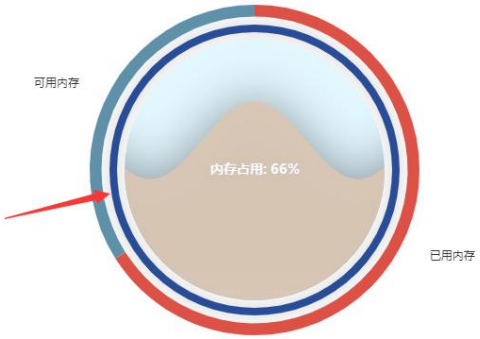
设置值大的时候展现就和山峰一样。



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。



参数较多，感兴趣的伙伴可以阅读 github 讲解。

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

以上就是 echarts 在 angular 中的三种使用方法，两种场景下对 echarts 的使用，希望可以帮助到大家！