Hostbinding 就是把内部的变量或函数绑定到渲染的 HTML 宿主元素上，从而改变其行为。比如

@HostBinding('class.hover') private ishovering: boolean;

这句就是把宿主元素的 hover 这个样式和成员变量 ishovering 绑定起来，ishovering 如果是 true，就应用这个样式，否就不应用。

当然除了样式，我们还可以绑定其他属性，比如我们做的动画中，我们这么写

@HostBinding('@routeAnim') state: string;

由于动画本身并没有可识别的属性，那么怎样才能让 state 的变化体现到宿主元素呢？这个 `@` 符号其实就是在宿主中创建了一个新的自定义属性 ng-trigger ng-trigger-routeAnim

<app-project-list

\_nghost-c15=""

class="ng-tns-c15-3

ng-trigger

ng-trigger-routeAnim"

style="display: flex; overflow: auto;">

</app-project-list>

HostBinding()可以为指令的宿主元素添加类、样式、属性

<https://blog.csdn.net/i042416/article/details/108950049>

@HostBinding()和@HostListener()在自定义指令时非常有用。@HostBinding()可以为指令的宿主元素添加类、样式、属性等，而@HostListener()可以监听宿主元素上的事件。

@HostBinding()和@HostListener()不仅仅用在自定义指令，只是在自定义指令中用的较多

本文基于Angular2+

下面我们通过实现一个在输入时实时改变字体和边框颜色的指令，学习@HostBinding()和@HostListener()的用法。

import { Directive, HostBinding, HostListener } from '@angular/core';

@Directive({

selector: '[appRainbow]'①

})

export class RainbowDirective{

possibleColors = [

'darksalmon', 'hotpink', 'lightskyblue', 'goldenrod', 'peachpuff',

'mediumspringgreen', 'cornflowerblue', 'blanchedalmond', 'lightslategrey'

];②

@HostBinding('style.color') color: string;

@HostBinding('style.borderColor') borderColor: string;③

@HostListener('keydown') onKeydown(){④

const colorPick = Math.floor(Math.random() \* this.possibleColors.length);

this.color = this.borderColor = this.possibleColors[colorPick];

}

}

说一下上面代码的主要部分：

①：为我们的指令取名为appRainbow

②：定义我们需要展示的所有可能的颜色

③：定义并用@HostBinding()装饰color和borderColor，用于设置样式

我们将directive内部的color属性值同host元素的css样式style.color建立了绑定关系。这样，用TypeScript修改directive的color属性值，会自动更新host元素的css样式值。

④：用@HostListener()监听宿主元素的keydown事件，为color和borderColor随机分配颜色

OK，现在就来使用我们的指令：

HostBinding和HostListener两个装饰器的理解和使用场景

https://www.jianshu.com/p/d9a89bb17e92

我的疑惑

不知道各位小伙伴在自学angular的时候有没有遇到过这两个装饰器——HostBinding和HostListener。这两个装饰器我在看官网的api说明的时候，真是百思不得其解，明明每个字我都能看懂，但是连在一起我就是看不懂了。而且官网上面给出的示例也很迷，让我搞不明白这俩api的使用场景到底是啥。

官网说明

先看看这两个api在官网上面的说明吧：

HostBinding（官网链接）：

用于把一个 DOM 属性标记为绑定到宿主的属性，并提供配置元数据。 Angular 在变更检测期间会自动检查宿主属性绑定，如果这个绑定变化了，它就会更新该指令所在的宿主元素。

HostListener（官网链接）：

用于声明要监听的 DOM 事件，并提供在该事件发生时要运行的处理器方法。

是不是看完之后也不知道这俩兄弟的使用场景到底是啥？在我看来，这俩装饰器的作用是为了方便我们将复杂的DOM操作可以抽取成一个指令，以此来精简代码。废话不多说，上代码，一看就懂。

代码示例

假设此时有一个业务场景，有一个输入框input，我们每次输入的时候，字的颜色和边框的颜色就会变化一下，我们取名叫“彩虹输入框”。

如果我们不使用组件封装的方式的话，直接在组件中写相关的DOM操作，代码如下：

@Component({

selector: 'app-rainbow-input-demo',

template: `

<h3>这是一个彩虹输入框，每输入一次都会改变颜色</h3>

<input [class]="'my-input'"

type="text"

[style]="inputStyleObj"

(keydown)="onKeyDown()"

/>

`,

styles:[

`.my-input {

border:none;

outline: none;

border: 2px solid #333333;

border-radius: 5px;

}`

]

})

export class RainbowInputDemoComponent {

//默认的颜色样式

public inputStyleObj = {

color:null,

borderColor:null,

};

//颜色库

public possibleColors = [

'darksalmon', 'hotpink', 'lightskyblue',

'goldenrod', 'peachpuff', 'mediumspringgreen',

'cornflowerblue', 'blanchedalmond', 'lightslategrey'

];

//键盘落下事件

onKeyDown(){

const index = Math.floor(Math.random() \* this.possibleColors.length);

//如果我们直接使用this.inputStyleObj.color = this.possibleColors[index]的话，

//this.inputStyleObj虽然内容变了，由于它是引用类型，其地址值没有变。所以不会触发视图的重新渲染

//在angular中，和react似的，我们直接修改引用类型不会触发重新渲染，只能覆盖它或者合并它，使其地址值发生改变，才会触发重新渲染

//如果觉得麻烦的话，完全可以在模板中使用[style.color]和[style.borderColor]

this.inputStyleObj = {

color:this.possibleColors[index],

borderColor:this.possibleColors[index],

}

}

}

效果如图：

彩虹输入框

就这样，我们实现了这个功能，那么现在有个问题。假如这种rainbowInput我们还需要在其他组件用到怎么办，难道每次使用的时候我们都把这些代码复制粘贴一遍吗？很明显这不符合组件封装的原则，如果你真的这么做，技术经理或者项目经理也会打爆你的狗头、

那么我们就需要将其封装成一个组件或者一个指令。在这篇文章中我们先把它封装成一个指令，至于原因后面再说。代码如下：

@Directive({

selector: '[appRainbow]'

})

export class RainbowInputDirective {

public possibleColors = [

'darksalmon', 'hotpink', 'lightskyblue',

'goldenrod', 'peachpuff', 'mediumspringgreen',

'cornflowerblue', 'blanchedalmond', 'lightslategrey'

];

//字体颜色

@HostBinding('style.color') color: string;

//边框颜色

@HostBinding('style.borderColor') borderColor: string;

//监听键盘落下的事件

@HostListener('keydown') onKeyDown() {

//获取一个随机的颜色

const index = Math.floor(Math.random() \* this.possibleColors.length);

this.color = this.borderColor = this.possibleColors[index];

}

}

@Component({

selector: 'app-rainbow-input-demo',

template: `

<h3>这是一个彩虹输入框，每输入一次都会改变颜色</h3>

<input [class]="'my-input'" type="text" appRainbow />

`,

styles:[

//省略，和上面相同

]

})

export class RainbowInputDemoComponent {}

就像上面代码写的那样，我们将重复的逻辑抽取了出来，使代码的维护性和美观度都有了很大的提升。

结论

在代码中我们可以看到@HostBinding的作用其实就是将某个属性绑定到了宿主元素上，但是这个属性不是随随便便的属性。这个属性指的是angular模板中支持的属性，其实@HostBinding就相当于模板中的[]或者bind-。同理@HostListener就相当于模板中的()或者on-。使我们可以在指令当中就可以将属性和方法绑定到宿主元素上，实现的效果和我们第一种直接将(keydow)和[style]直接写在模板上是一样的。所以说，这俩装饰器里面的字符串那可不能随便写。

关于指令和组件

但是，其实在angular当中，组件和指令其实区别不是特别大，因为angular中的组件装饰器@Component就是继承自@Directive的。

其实我们将这个DOM操作封装成一个组件也不是不可以，代码如下

@Component({

selector:'input[appRainbow]'

})

但是和指令写法：

@Directive({

selector: '[appRainbow]'

})

真的差别不大。

总结

@HostBinding相当于模板上的[]或者bind-；

@HostListener相当于模板上的()或者on-；

就是angular为了不希望我们直接操作DOM，提供的指令中的数据和方法绑定。