angular学习

汪洋

2017 年 9 月 11 日

目录

1 Pipe 管道

只有当string,number值改变,或者Object的引用发生改变,才会触发管道

@Pipe({ name: 'pipeName', pure: false })

将管道声明为不纯的管道,不纯的管道可以监听对象属性的改变,但是速度会更慢.

2 Animation 动画

2.1 configure

@Component({ animations:[] })

组件注解中添加额外的属性 animations {

```
trigger('animationName',[
  state('state1', style({})),
  state('state2', style({})),
  transition('state1 => state2', animation('100ms ease-out'))
]) }
```

transition 有三个方向 =>, <= 和 <=>

state有三种 void, * 和自定义

1. void 表示当刚开始挂载到Dom节点或者从Dom移除时
2. * 表示任意state
3. state 自定义的state

void => * 的别名是 :enter

* => void 的别名是 :leave

如果多个状态之间的变化是相同的时间和特效.可以将状态变化写在一起,使用,分开.

状态变化的样式也可以内联到transition中

```
transition('inactive => active', [
  style({
    backgroundColor: '#cfd8dc',
    transform: 'scale(1.3)'
```

```
  }),
  animate('80ms ease-in', style({
    backgroundColor: '#eee',
    transform: 'scale(1)'
  })) ])
```

style:中的属性单位有 px, em 和%,当只提供数字而不提供单位时,angular会使用默认的单位px,当不知道具体的数字时,可以使用*号代替,angular会自动计算.

timing delay easing , timing easing 是animation的属性,即运行时间,延迟时间以及动画特效.

2.2 keyFrame

```
animate(300,keyframes[
  style({...styles,offset:0}),
  style({...styles,offset:1})
]);
```

offset 是动画的时间是一个小数(相对整体时间的百分比).这个属性是可选的,当没有时,会按照keyframes的数量来平均分配时间.

2.3 callbacks,start,done

```
(@animationName.start)
(@animationName.done)
[@animationName]="'initState'"
```

3 Forms

3.1 user input

3.1.1 Input

```
(keyup)="onKey($event)"
onKey(event: any){
}
onKey(event: KeyboardEvent){
```

((<HTMLInputElement>)event.target)
}
<input #name />
{{name.value}}
<input #name (keyup)="onKey(name.value)" />
{{nameValue}}
<input #nameInput (keyup.enter)="onKey(nameInput.value)" />
{{name}}
//keyup.enter 是对键盘Enter键按下,按起之后
<input #nameInput (keyup.enter)="update(nameInput.value)" (blur)="update(nameInput.value)" />
{{name}}

3.2 form,template-driven,reactive-form

FormControl 有 value, status, pristine, untouched status 是 VALID, INVALID, PENDING, or DISABLED值中的一个 pristine 是和 dirty 取反

3.2.1 template-driven

1. 需要引用 FormModule
2. 输入双向绑定 [(ngModel)]
3. name 属性,angular Form需要一个name来注册
4. angular

将自动添加3种类(即 className) ng-(un)touched, ng-(dirty,pristine), ng-(in)valid

内建的几个验证类

1. required
2. minlength="4"

自定义验证方法

1. 工厂方法

   export function forbiddenNameValidator(nameRe: RegExp): ValidatorFn {

   return (control: AbstractControl): {[key: string]: any} => {

   const forbidden = nameRe.test(control.value);

   return forbidden ? {'forbiddenName': {value: control.value}}: null;

   };

   } }

2. 指令

   @Directive({selector: '[forbiddenName]', providers: [{provide: NG_VALIDATORS, useExisting: ForbiddenValidatorDirective, multi: true}]})

   export class ForbiddenValidatorDirective implements Validator {

   @Input() forbiddenName: string;

   validate(control: AbstractControl): {[key: string]: any} {

   return this.forbiddenName ? forbiddenNameValidator(new RegExp(this.forbiddenName, 'i'))(control): null;

   }}

   推荐将数据输入,数据展示分开

3.2.2 modal-driven

ngOnInit(): void {
this.heroForm = new FormGroup({
'name': new FormControl(this.hero.name,
[Validators.required, Validators.minLength(4), forbiddenNameValidator(/bob/i) // <—— Here's how you pass in the custom validator.
]),
'alterEgo': new FormControl(this.hero.alterEgo),
'power': new FormControl(this.hero.power, Validators.required)
})};

3.2.3 注意

1. 还没有写代码测试
2. 由于template-driven 基于directive,所以可能会导致渲染周期会不一致
3. 由于是基于modal的,必须等全部渲染完成才能和用户进行交互?