angular学习

汪洋

2017 年 9 月 9 日

目录

1 Structural Directives

1.1 *ngIf

```
<div *ngIf=”hero”>{{hero.name}}</div>
```

是一个语法糖,将被翻译成

```
<div template=”ngIf hero”>{{hero.name}}</div>
```

接下来会生成

```
<ng-template [ngIf]=”hero”>
  <div>{{hero.name}}</div>
</ng-template>
```

1.2 *ngFor

```
<div *ngFor=”let hero of heroes;let i = index;let odd=odd;trackBy: trackById” [class.odd]=”odd”>
{{hero.name}}
</div>
```

将被翻译成

```
<div template=”ngFor let hero of heroes; let i=index; let odd=odd; trackBy: trackById” [class.odd]=”odd”>
    ({{i}}) {{hero.name}}
</div>
```

将生成

```
<ng-template ngFor let-hero [ngForOf]=”heroes” let-i=”index” let-odd=”odd” [ngForTrackBy]=”trackById”>
    <div [class.odd]=”odd”>({{i}}) {{hero.name}}</div>
</ng-template>
```

*ngFor 有几个内置的变量 index odd $implicit

上面的hero由于没有值,所以使用$implicit初始化了

在*ngFor中声明的变量只存在于每个重复的实例中,在其他位置仍然可以声明相同名称的变量

其他的几个内置的结构指令和*ngFor,*ngIf的展开方式一样

1.3 ngSwitch

ngSwitch 本省并不是一个结构指令,它只是给*ngSwitchCase提供一个值.

```
<div [ngSwitch]=hero?.emotion>
<happy–hero *ngSwitchCase=happy [hero]=hero></happy–hero>
<sad–hero *ngSwitchCase=sad [hero]=hero></sad–hero>
<confused–hero *ngSwitchCase=confused [hero]=hero></confused–hero>
<unknown–hero *ngSwitchDefault [hero]=hero></unknown–hero>
</div>
```

将被翻译成

```
<div [ngSwitch]=”hero?.emotion”>
<happy–hero template=”ngSwitchCase ’happy’” [hero]=”hero”></happy–hero>
<sad–hero template=”ngSwitchCase ’sad’” [hero]=”hero”></sad–hero>
<confused–hero template=”ngSwitchCase ’confused’” [hero]=”hero”></confused–hero>
<unknown–hero template=”ngSwitchDefault” [hero]=”hero”></unknown–hero>
</div>
```

将生成

```
<div [ngSwitch]=”hero?.emotion”>
<ng–template [ngSwitchCase]=”’happy’”>
<happy–hero [hero]=”hero”></happy–hero>
</ng–template>
<ng–template [ngSwitchCase]=”’sad’”>
<sad–hero [hero]=”hero”></sad–hero>
</ng–template>
<ng–template [ngSwitchCase]=”’confused’”>
<confused–hero [hero]=”hero”></confused–hero>
</ng–template >
<ng–template ngSwitchDefault>
<unknown–hero [hero]=”hero”></unknown–hero>
</ng–template></div>
```

1.4 自定义结构指令

1. 构造函数有2个注入的参数
   - (a) TemplateRef
   - (b) ViewContainerRef
2. enumerate

1.5 注意

angular 不允许同时在同一个元素中使用多个结构指令

ng-template是angular内建的元素,如果它不与angular的指令配合使用的话,将被angular从Dom节点中移除

ng-container是angular内建的元素,但是与ng-template不同的是,它相当于一个语句块,本身不会显示出来,但是内部的其他符合结构指令条件的元素会显示出来

2 Pipes

管道使用|来标志

angular本身提供了全局的管道

DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe, PercentPipe

2.1 参数

管道使用colon:来接受参数,如果有多个参数,则使用多个:

|date:'MM/dd/yy'

|slice:1:5

2.2 Chaining pipes,管道链

2.3 自定义管道

1. 使用@Pipe,有一个name属性,以后可能会有pure属性
2. 继承PipeTransform,transform方法至少一个参数value,然后是额外的参数

2.4 注意

1. 使用管道是昂贵的.每一次键盘事件,鼠标移动,时钟滴答,以及服务器响应后都会执行一次脏检查.

3 零碎

3.1 ngModal

1. 必须引入FormsModule