Detect when input value changed in directive

Asked 2 years, 9 months ago Active 2 months ago Viewed 39k times

30

1

I'm trying to detect when the **value** of an input changed in a directive. I have the following directive:

```
import { ElementRef, Directive, Renderer} from '@angular/core';

@Directive({
    selector: '[number]',
    host: {"(input)": 'onInputChange($event)'}
})

export class Number {

    constructor(private element: ElementRef, private renderer: Renderer){

    }
    onInputChange(event){
        console.log('test');
    }
}
```

The problem in this directive is that it detects only when there is an input and not when the value changes programatically. I use reacive form and sometimes I set the value with the `patchValue()` function. How can I do so the change function gets triggered?

angular angular-directive angular2-directives

asked Jan 18 '17 at 19:39

ncohen
2,615 13 55 97

3 Answers

46

✓

```
@Directive({
    selector: '[number]'
})
export class NumberDirective implements OnChanges {
    @Input() public number: any;
    @Input() public input: any;

    ngOnChanges(changes: SimpleChanges){
      if(changes.input){
        console.log('input changed');
      }
    }
}
```

Plunkr

Stackblitz

edited Jul 18 at 15:32

answered Jan 18 '17 at 20:25

Teddy Sterne
8,600 1 23 35

1 ngOnChanges(changes : SimpleChanges) – Carlos Bravo Jun 2 '17 at 10:28

This requires an input directive, it should use host listener without any dependencies – evanjmg Jan 8 '18 at 15:49

why this solution doesn't work in Angular 6? – GlacialMan Jul 12 at 8:03

@GlacialMan This still works for me in Angular 8 – Teddy Sterne Jul 18 at 15:33

hmm, okay maybe I don't know how to implement it :/ – GlacialMan Jul 19 at 7:52

3

There is a better way to use this result, used for example in the `*ngIf` Angular source code.

You can combine an `@Input()` with a `setter`. When the input changes, the setter is called again.

```
    }
    else if(wheels === 4) {
      console.log("It's a car!");
    }
    else {
      console.log("I don't know what it is :(");
    }
  }
```

You can save the previous value in the Directive property in order to use it later and compare it with the new value:

```
private previousValue: any = null;

@Input() set myInputName(value: any) {
  console.log(`Previous value is: ${this.previousValue}`);
  console.log(`New value is: ${value}`);
  this.previousValue = value;
}
```

answered Apr 27 '18 at 0:49
Cristian Traìna
4,229 1 16 39

You are not answering to the question asked – Mattew Eon Feb 7 at 9:04

@MattewEon Wtf? This solves perfectly the problems, since it detects when the input is changed programmatically – Cristian Traìna Feb 7 at 9:08

The goal isn't to add a new `@Input` property to the component, he want to detect the change on the ngModel property – Mattew Eon Feb 7 at 9:28

Good approach using latest Angular 7 @Input style. – SushiGuy Feb 11 at 22:31

2

You can also use HostListener. For more information about HostListener, you can go through this link. Here is the code.

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


@Directive({
       selector: '[number]'
```

```
    @Input() public number: any;
    @Input() public input: any;

    constructor(private el: ElementRef) {}

    @HostListener('change') ngOnChanges() {
        console.log('test');
    }

}
```

answered Jun 16 '17 at 10:41

Tushar Ghosh
434 6 15

2 'change' not work, 'mouseenter', 'mouseleave' from angular sample work. Please, test your answers. – bmi Jul 11 '18 at 13:03

1 'change' works for select menus, at least (as does 'ngModelChange', and you can get the new value by injecting the `$event` var: `@HostListener('ngModelChange', ['$event']) public doStuff(value) { ... }`) – mopo922 Mar 26 at 19:50