# Angular Best Friends

## Module 3 Demo 2 – Change detection

## Goals

Purpose of this demo is to show how change detection strategies can be applied when using the container presentation pattern. This is very useful since in this pattern we normally don’t want to let presentation components change the data without user input. So to guard against changes of data in the code of presentation components we can implement a change detection strategy.

This demo will be done based on the code resulted from “Module 3 Exercise 1 – Input and Output”.

## Steps

1. Go to the “customer-details.component.ts” file. Here we want to simulate data changing through code (so not through user input). So we’ll manipulate the data in the ngOnInit() function.

ngOnInit() {

this.customer.name = "Tina Doe";

setTimeout(()=> {

this.customer.name = "SomeOther Doe";

}, 2000);

}

1. After doing this we’ll notice that the child component simply changes the customer name after 2 seconds directly through code, without any customer input. This is certainly not desired in this pattern. Presentation components should not change data in code. Data should be changed only through user input. So we have to do something about it!
2. In the component declaration we can define a change detection strategy.

@Component({

selector: 'app-customer-details',

templateUrl: './customer-details.component.html',

styleUrls: ['./customer-details.component.css'],

changeDetection: ChangeDetectionStrategy.OnPush

})

1. Now refresh the browser. You’ll notice that the initial value remains there for the customer’s name until you press the button. So the change detection strategy prevents the changes coming from code to be reflected.