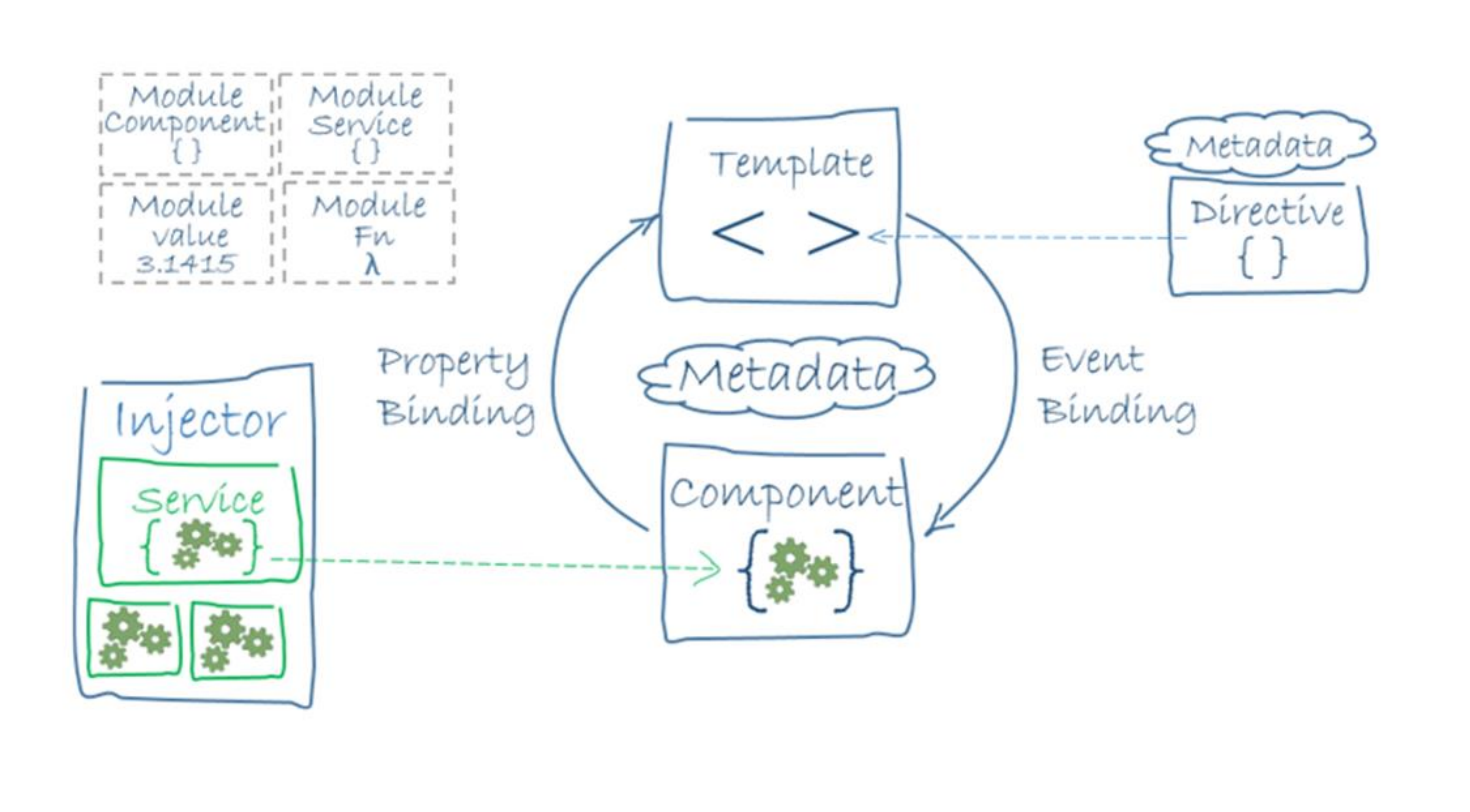
ROAD SHOW

Microsoft Student Partners

# *Decorators and inheritance in TypeScript*

Speaker: Anton Dolhopolov

# *Angular architecture*



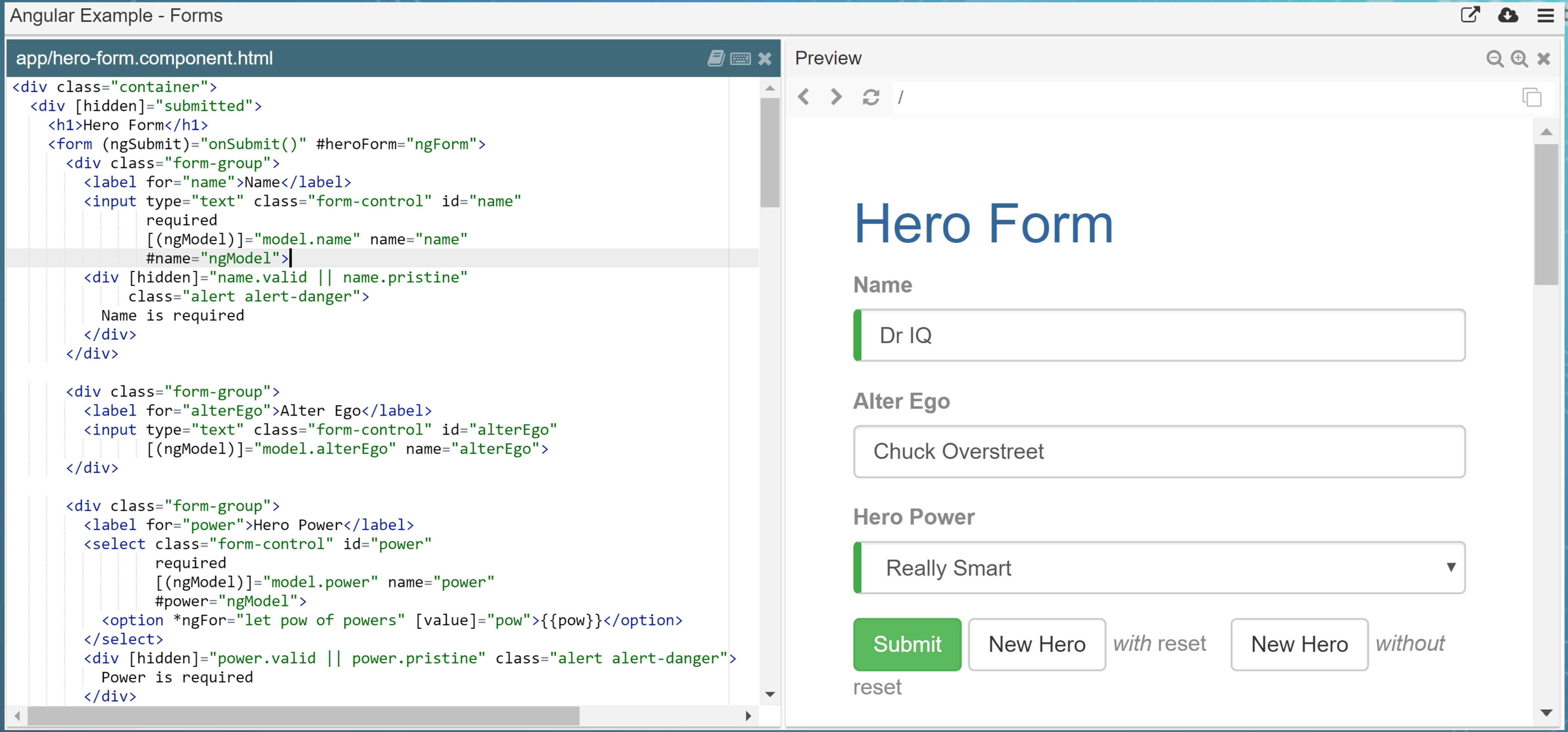
# *Forms in Angular*

1. Template-driven forms

2. Reactive forms

3. Dynamic forms

# Template-driven forms



Angular Example - Forms

app/hero-form.component.html

```html
<div class="container">
  <div [hidden]="submitted">
    <h1>Hero Form</h1>
    <form (ngSubmit)="onSubmit()" #heroForm="ngForm">
      <div class="form-group">
        <label for="name">Name</label>
        <input type="text" class="form-control" id="name"
               required
               [(ngModel)]="model.name" name="name"
               #name="ngModel">
        <div [hidden]="name.valid || name.pristine"
             class="alert alert-danger">
          Name is required
        </div>
      </div>

      <div class="form-group">
        <label for="alterEgo">Alter Ego</label>
        <input type="text" class="form-control" id="alterEgo"
               [(ngModel)]="model.alterEgo" name="alterEgo">
      </div>

      <div class="form-group">
        <label for="power">Hero Power</label>
        <select class="form-control" id="power"
                required
                [(ngModel)]="model.power" name="power"
                #power="ngModel">
          <option *ngFor="let pow of powers" [value]="pow">{{pow}}</option>
        </select>
        <div [hidden]="power.valid || power.pristine" class="alert alert-danger">
          Power is required
        </div>
```

Preview

/

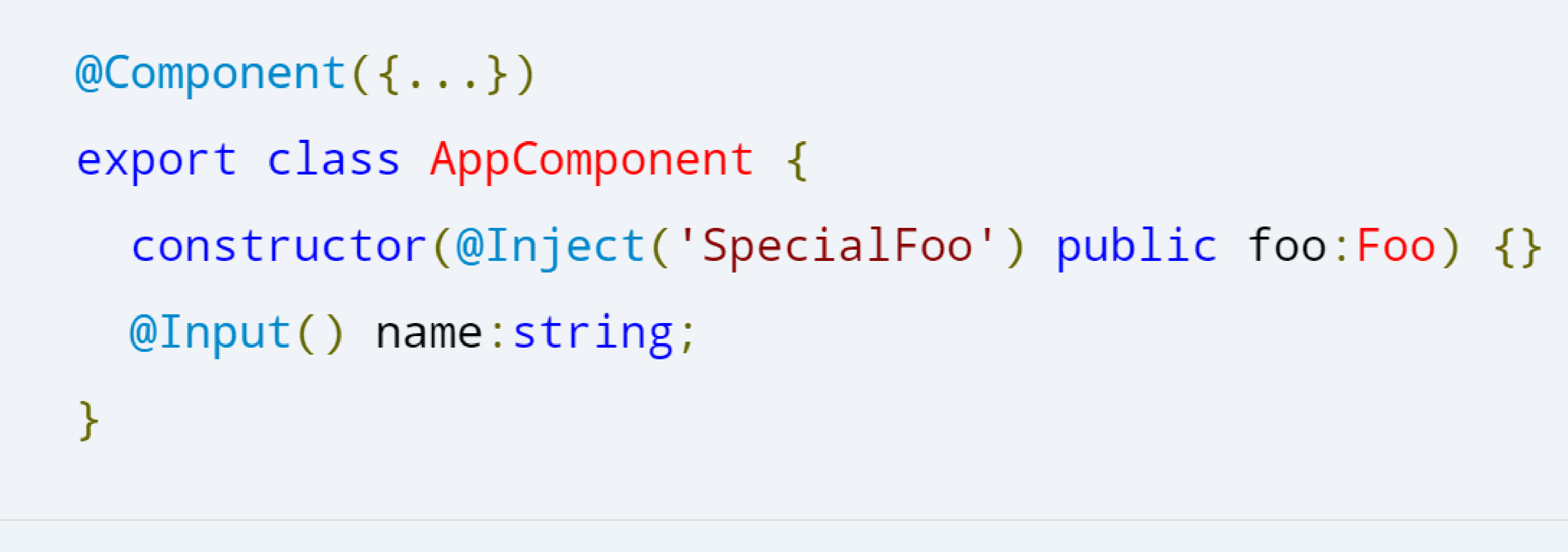
## Hero Form

Name: Dr IQ

Alter Ego: Chuck Overstreet

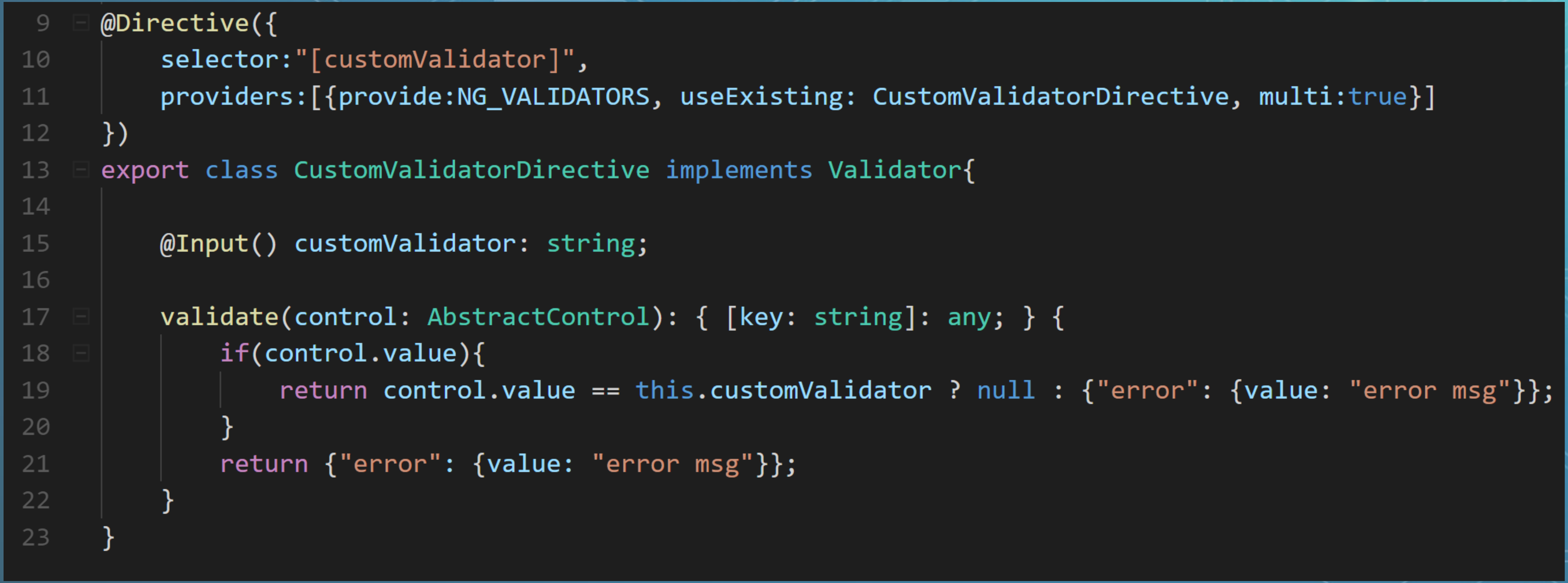
Hero Power: Really Smart ▾

# Decorators



```
@Component({...})
export class AppComponent {
  constructor(@Inject('SpecialFoo') public foo:Foo) {}
  @Input() name:string;
}
```

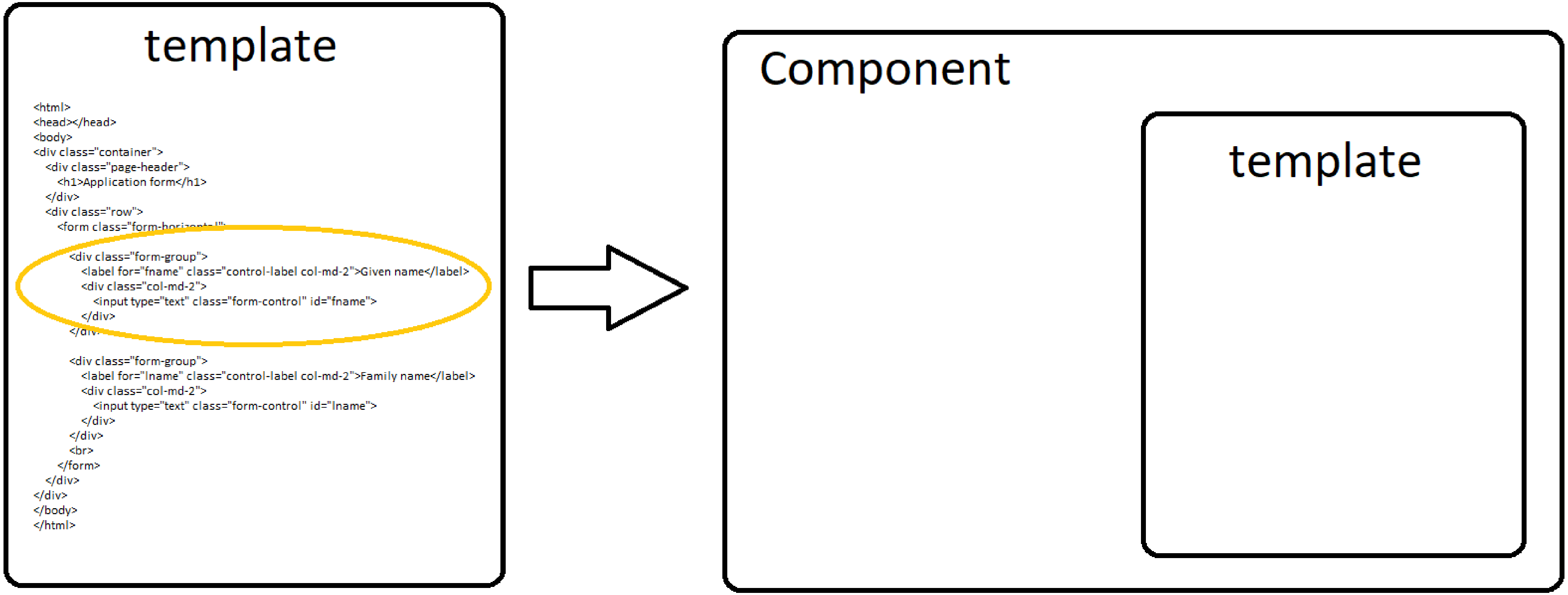
# Custom validation



```
@Directive({
    selector:"[customValidator]",
    providers:[{provide:NG_VALIDATORS, useExisting: CustomValidatorDirective, multi:true}]
})
export class CustomValidatorDirective implements Validator{

    @Input() customValidator: string;

    validate(control: AbstractControl): { [key: string]: any; } {
        if(control.value){
            return control.value == this.customValidator ? null : {"error": {value: "error msg"}};
        }
        return {"error": {value: "error msg"}};
    }
}
```

# Don't repeat yourself

Microsoft Student Partners

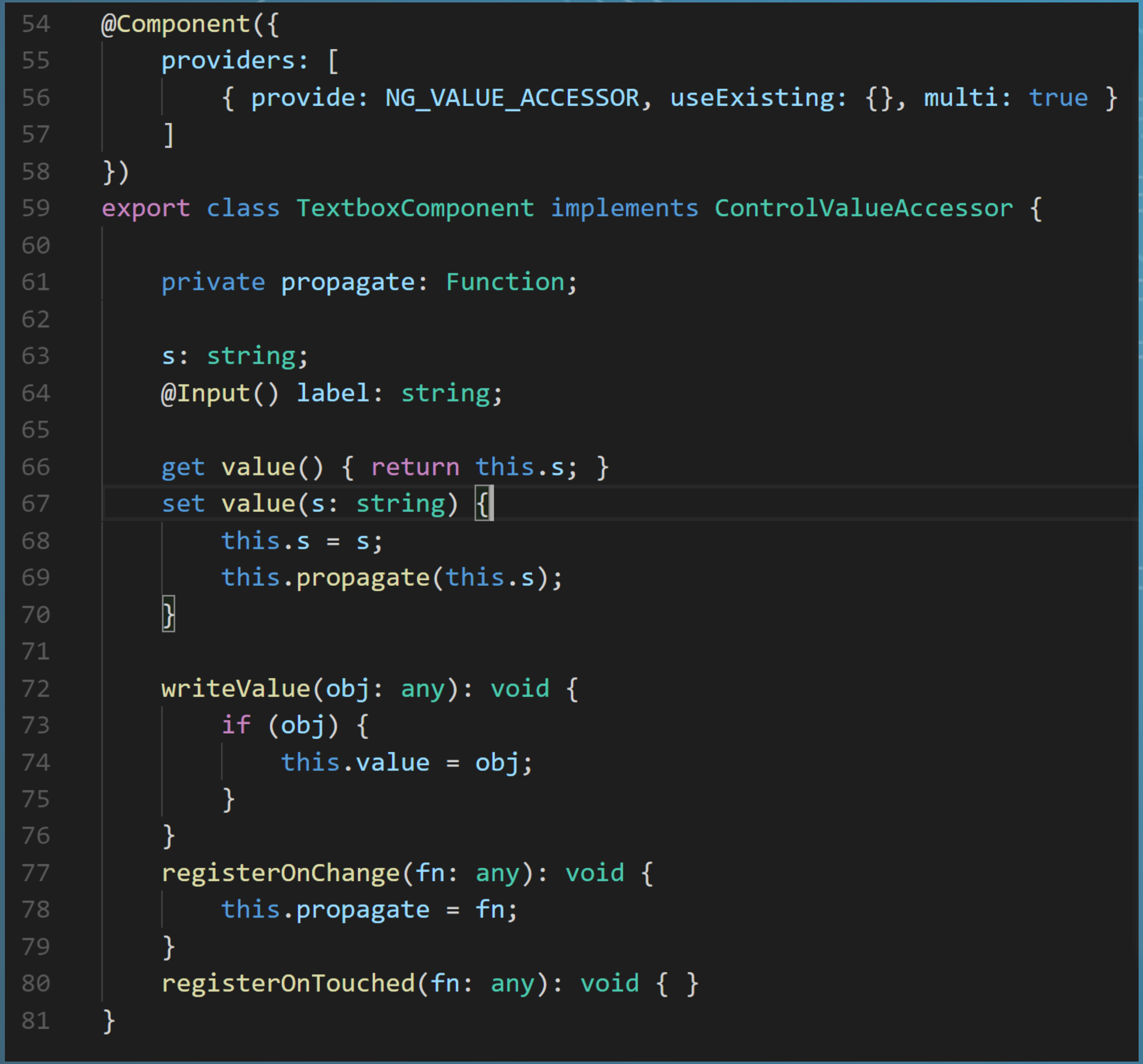


```
<html>
<head></head>
<body>
<div class="container">
  <div class="page-header">
    <h1>Application form</h1>
  </div>
  <div class="row">
    <form class="form-horizontal">

      <div class="form-group">
        <label for="fname" class="control-label col-md-2">Given name</label>
        <div class="col-md-2">
          <input type="text" class="form-control" id="fname">
        </div>
      </div>

      <div class="form-group">
        <label for="lname" class="control-label col-md-2">Family name</label>
        <div class="col-md-2">
          <input type="text" class="form-control" id="lname">
        </div>
      </div>
      <br>
    </form>
  </div>
</div>
</body>
</html>
```

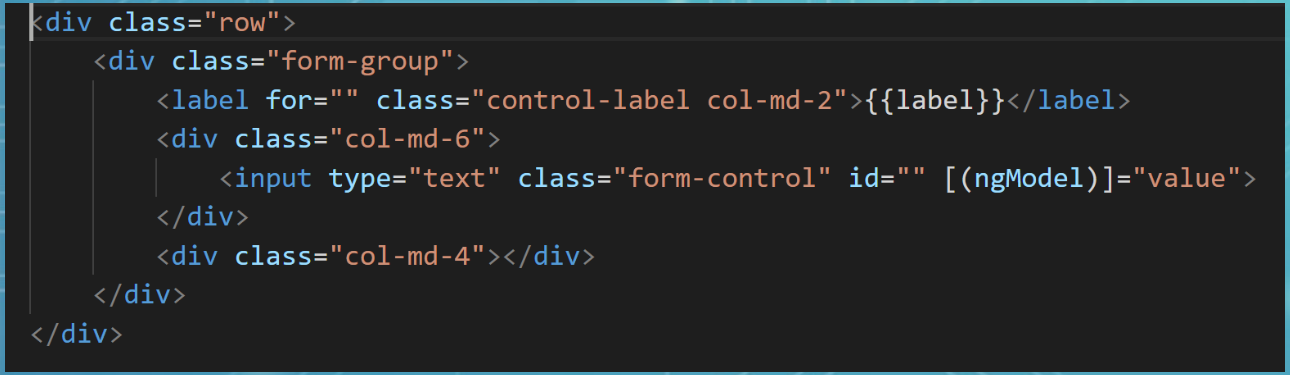
# 2-way data binding

ROAD SHOW

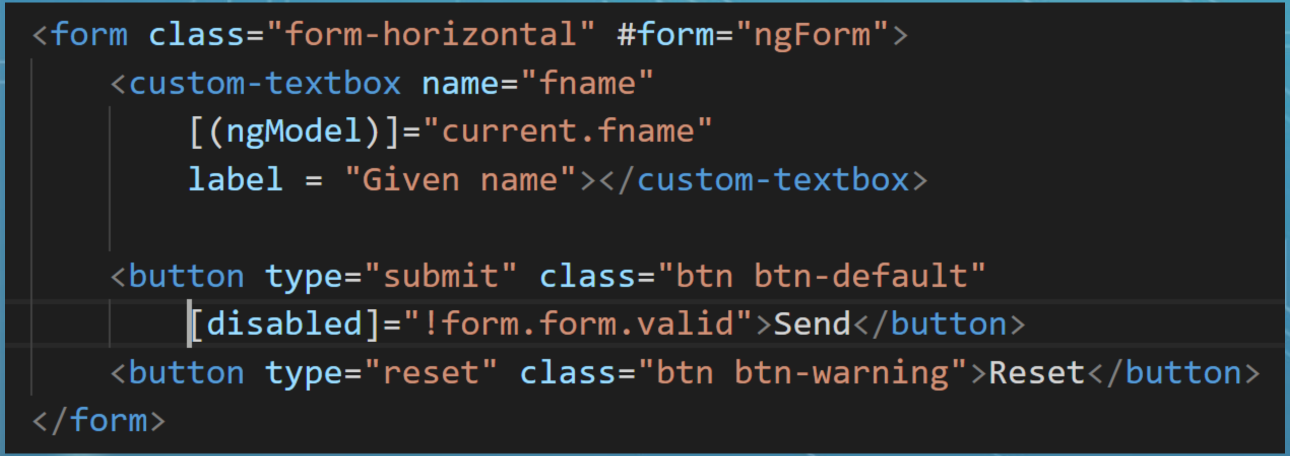
Microsoft Student Partners



```
@Component({
    providers: [
        { provide: NG_VALUE_ACCESSOR, useExisting: {}, multi: true }
    ]
})
export class TextboxComponent implements ControlValueAccessor {

    private propagate: Function;

    s: string;
    @Input() label: string;

    get value() { return this.s; }
    set value(s: string) {
        this.s = s;
        this.propagate(this.s);
    }

    writeValue(obj: any): void {
        if (obj) {
            this.value = obj;
        }
    }
    registerOnChange(fn: any): void {
        this.propagate = fn;
    }
    registerOnTouched(fn: any): void { }
}
```

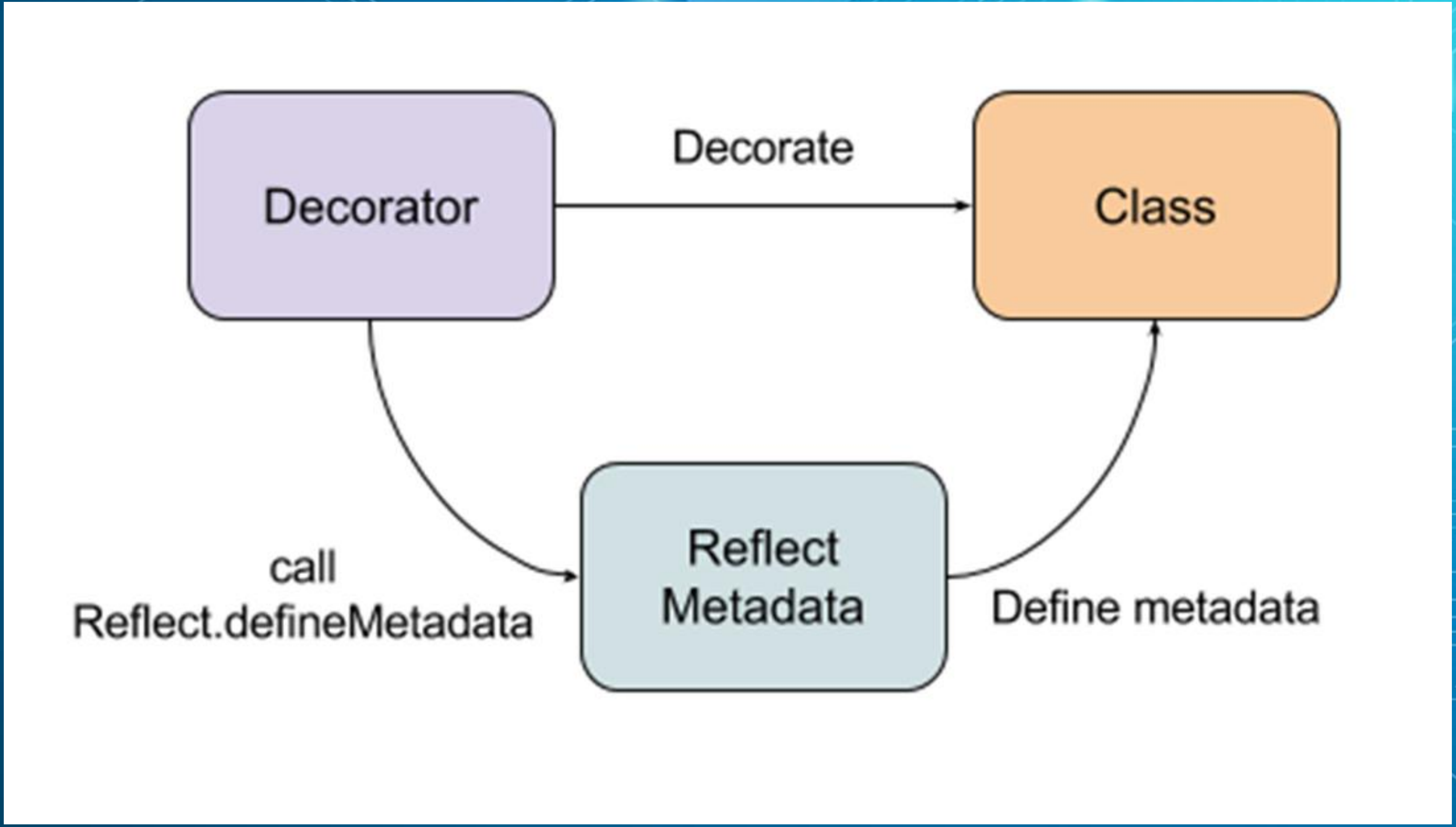


```
<div class="row">
    <div class="form-group">
        <label for="" class="control-label col-md-2">{{label}}</label>
        <div class="col-md-6">
            <input type="text" class="form-control" id="" [(ngModel)]="value">
        </div>
        <div class="col-md-4"></div>
    </div>
</div>
```



```
<form class="form-horizontal" #form="ngForm">
    <custom-textbox name="fname"
        [(ngModel)]="current.fname"
        label = "Given name"></custom-textbox>

    <button type="submit" class="btn btn-default"
        [disabled]="!form.form.valid">Send</button>
    <button type="reset" class="btn btn-warning">Reset</button>
</form>
```

# Decorators



Decorator

Decorate

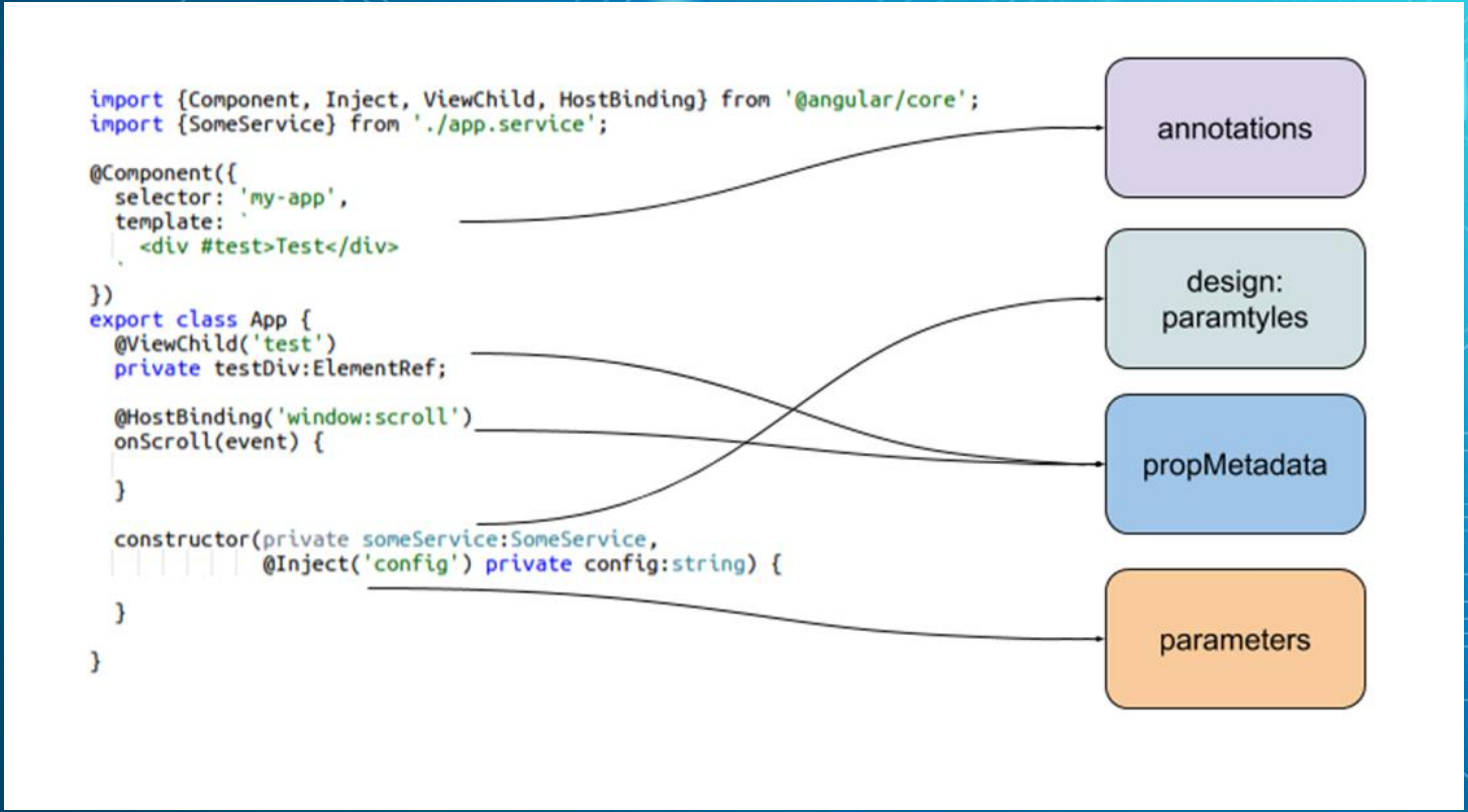
Class

call
Reflect.defineMetadata

Reflect
Metadata

Define metadata

# Decorators



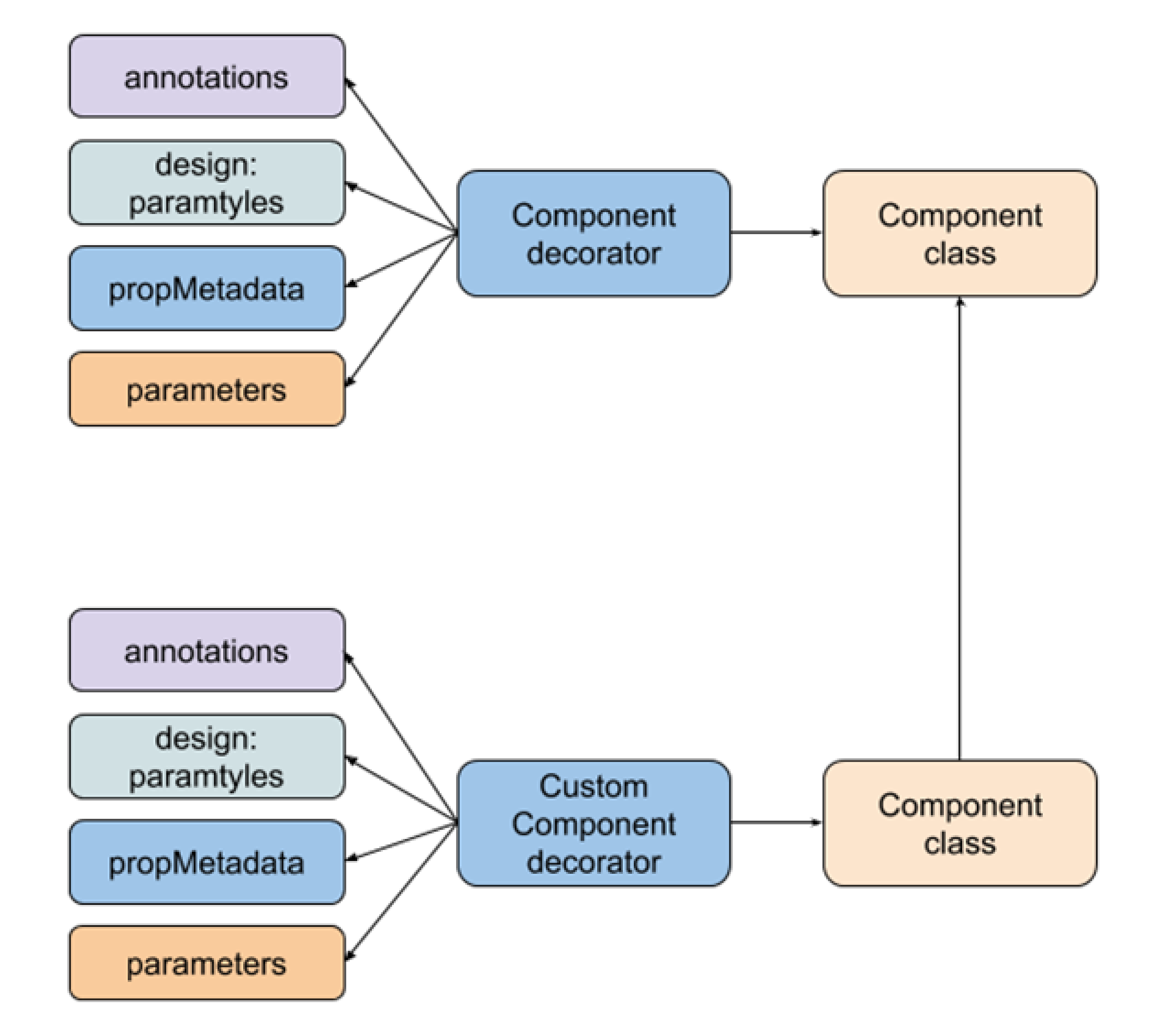
```
import {Component, Inject, ViewChild, HostBinding} from '@angular/core';
import {SomeService} from './app.service';

@Component({
  selector: 'my-app',
  template: `
    <div #test>Test</div>
  `
})
export class App {
  @ViewChild('test')
  private testDiv:ElementRef;

  @HostBinding('window:scroll')
  onScroll(event) {

  }

  constructor(private someService:SomeService,
              @Inject('config') private config:string) {

  }

}
```

# *What about inheritance?*

ROAD SHOW
Microsoft Student Partners

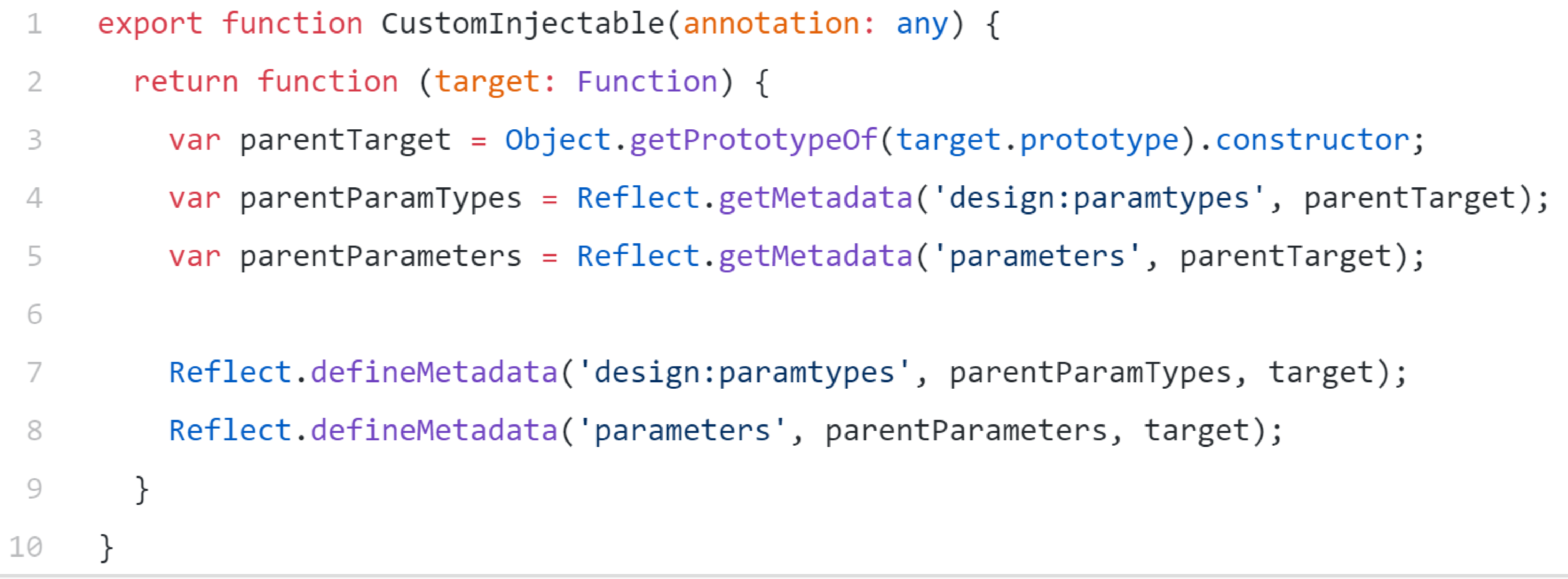
# *Everything is good. Almost.*

# *Inheritance*



# Custom decorator

Microsoft Student Partners



```
export function CustomInjectable(annotation: any) {
  return function (target: Function) {
    var parentTarget = Object.getPrototypeOf(target.prototype).constructor;
    var parentParamTypes = Reflect.getMetadata('design:paramtypes', parentTarget);
    var parentParameters = Reflect.getMetadata('parameters', parentTarget);

    Reflect.defineMetadata('design:paramtypes', parentParamTypes, target);
    Reflect.defineMetadata('parameters', parentParameters, target);
  }
}
```

## References:

1. https://angular.io/docs

2. https://medium.com/@ttemplier/angular2-decorators-and-class-inheritance-905921dbd1b7

3. https://medium.com/@tarik.nzl/angular-2-custom-form-control-with-validation-json-input-2b4cf9bc2d73

4. https://blog.thoughtram.io/angular/2016/07/27/custom-form-controls-in-angular-2.html

# *Thanks for your time!*

# P.S. demo materials is on

https://github.com/BrodaUa/msp/tree/master/roadshow2017