GFT

Angular components

GFT INTERNAL TRAINING

INNOVATE. TRANSFORM. DELIVER.

Agenda

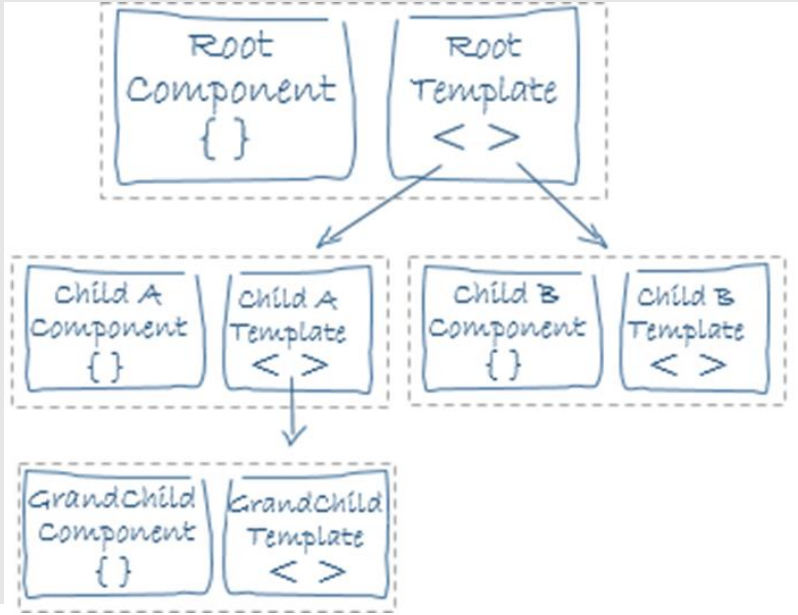
1.component

2.pasing data to a component

component

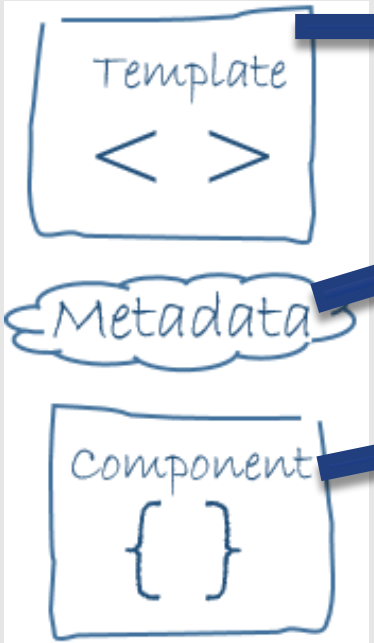
component

- controls a patch of screen real estate that we could call a view, and declares reusable UI building blocks for an application
- is anything that is visible to the end user and which can be reused many times within an application
- is a core concept
- app → tree of components



component

.




```
<p>foo works!</p>
```

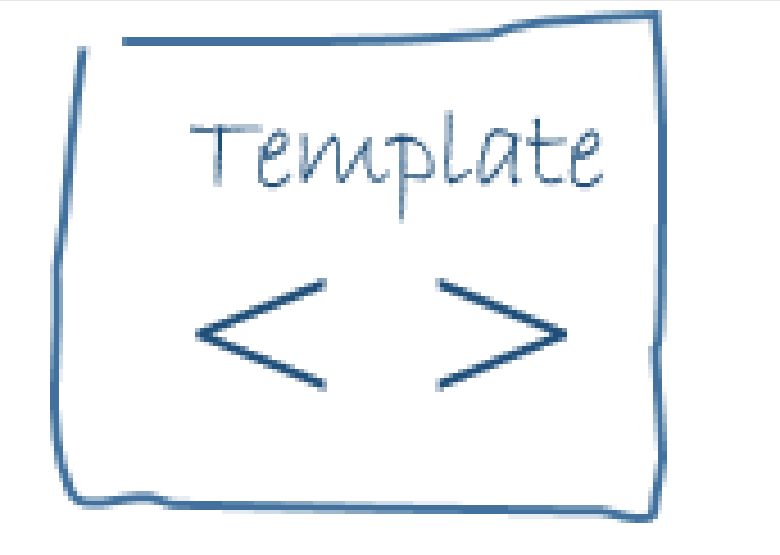
(foo.component.html)

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-foo',
  templateUrl: './foo.component.html',
  styleUrls: ['./foo.component.css']
})
export class FooComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

(foo.component.ts)

template syntax is not html

.



https://angular.io/guide/template-syntax

Pasing data to a component

Top-down (from parent to child)

Top-down (from parent to child)

```
import {Input, Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-saluda-a',
  template: 'Hola {{name}}',
})
export class SaludaAComponent  {
  @Input() name: string;

}
```

CHILD COMPONENT

Top-down (from parent to child)

```
@Component({
  selector: 'app-root',
  template:
   `<h1>Father component</h1>
    <!-- property binding to a string -->
    <app-saluda-a name="Nelson Mandela"></app-saluda-a>
    <br><br>
    <!-- property binding to an expression  -->
    <app-saluda-a [name]="famous.toUpperCase()"></app-saluda-a>
    <br>`
})
export class AppComponent {
  famous = "rafa nadal";
}
```

PARENT COMPONENT

Top-down (from parent to child)

property binding

```
`<h1>Father component</h1>
 <!-- property binding to a string -->
 <app-saluda-a name="Nelson Mandela"></app-saluda-a>
 <br><br>
 <!-- property binding to an expression  -->
 <app-saluda-a [name]="famous.toUpperCase()"></app-saluda-a>
```

Father component

Hola Nelson Mandela

Hola RAFA NADAL

Property binding is dynamic

```
    <!-- property binding is dynamic -->
    <app-saluda-a [name]="changingName"></app-saluda-a>
    <br>`
})
export class AppComponent {
  famous = "rafa nadal"
  changingName ="ben alpert"

  constructor(){
    setTimeout( () => this.changingName = "sophie alpert", 2000)
  }
```

Bottom-up (from child to parent)

Bottom-up (from child to parent)

```
import {Input, Output, Component, EventEmitter, OnInit } from '@angular/core';

@Component({
  selector: 'app-saluda-a',
  template: `<p>Hola {{name}}</p>
            <button (click)=confirmRead()>Message readed!!</button>`,
})
```

```
@Output() readed = new EventEmitter<string>();

confirmRead = function(){
  this.readed.emit("Message has been readed");
}
```

CHILD COMPONENT

Bottom-up (from child to parent)

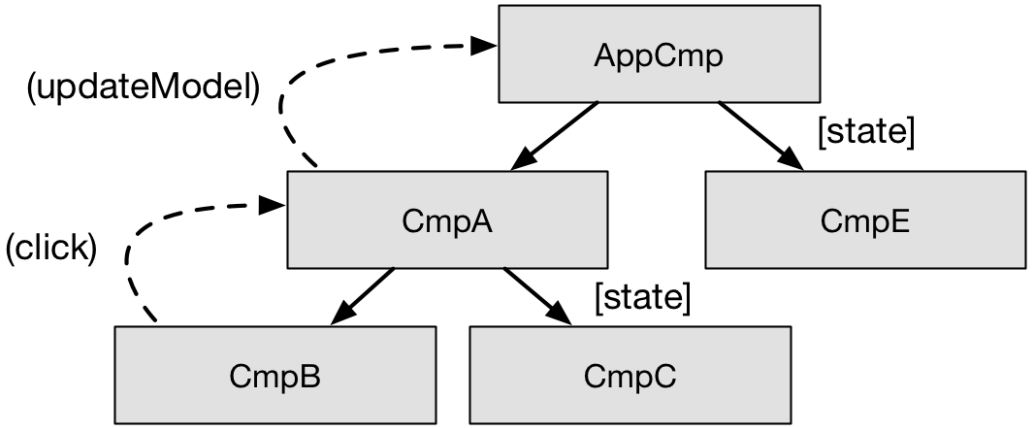
event binding

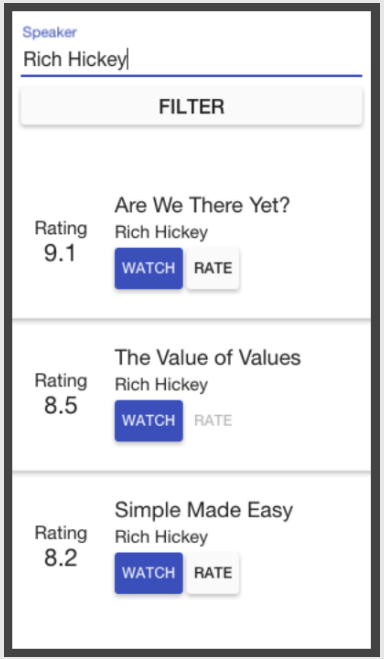
```
<h1>Father component</h1>
<!-- property binding to a string -->
<app-saluda-a name="Nelson Mandela" (readed)=doSomething($event)></app-saluda-a>
```

```
doSomething = function(message)  {
  alert(message);
}
```

PARENT COMPONENT

change propagation through the component tree





https://blog.nrwl.io/reactive-programming-in-angular-7dcded697e6c

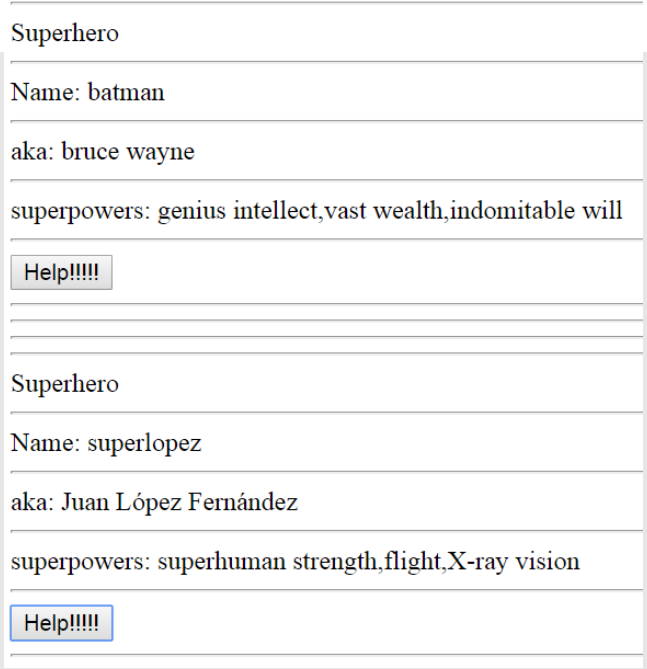
learn by doing

```
export class Superhero {
  name: string;
  alterEgo: string;
  superpowers: string[];
}
```

create an angular app with a component showSuperhero that receives from a parent component a Superhero object and show his data

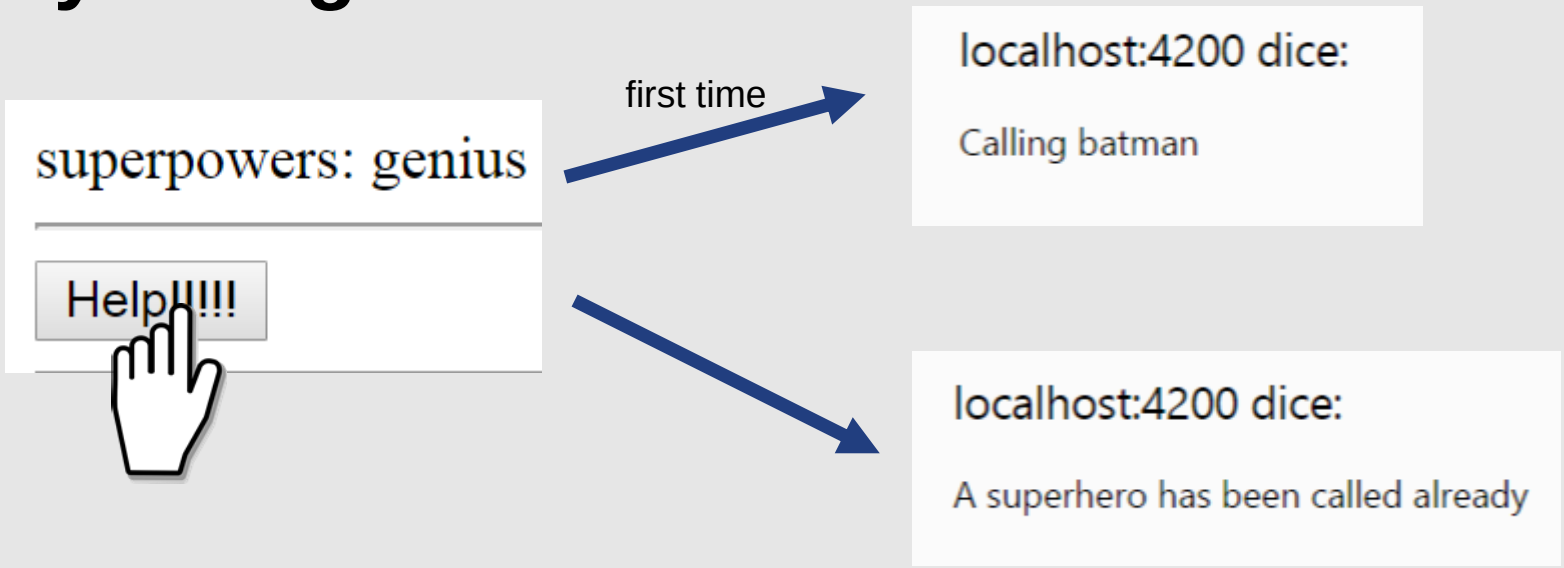
Instead of class an interface could have been used, more on the differences here:
https://stackoverflow.com/questions/40973074/difference-between-interfaces-and-classes-in-typescript

learn by doing



add a button to showSuperhero component that will send to its parent the name of the superhero

learn by doing



show an alert calling a superhero only once

learn by doing

- use *ngFor to add the components from a superheroes lists
- allow to call as many heroes as the user wants
- show a table with the status of each superhero (on duty or / available)
- add a setTimeout with a random value so each superhero can comunicate when is available again
- highlight with style the components that are on duty

Shaping the future of digital business

GFT Internal Technical Training

Eduardo García Ibaseta

eduardo.garcia-ibaseta@gft.com
+34 935 639476

GFT IT Consulting, S.L.
Av. Alcalde Barnils, 69-71
08174 Sant Cugat del Vallès (BARCELONA)