Interpolación y property data binding

src/app/talks/ = talks.component.css, talks.component.html ⇒ TEMPLATE, talks.component.spec.ts, talks.component.ts ⇒ COMPONENTE

ng generate component talks

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

@Component({
  selector : 'app-talks',
  template : 'talks funciona',
})

export class TalksComponent implements OnInit {
  constructor() {
  }
  ngOnInit() {
  }
}
```

Uso del componente por selector

Ej. app.component.ts ?

↳ template 4

<app-talks> <app-talks>

Interpolación

En template: {{title}}

En constructor(){
this.title = "Hola";
setTimeout(() =>
this.title = 'Angular charges!');
}

Reflejo del valor de forma estática en 3000 segundos

Forma dinámica
Evento DOM, DDDD, etc

esquelete

sintaxis

Interpolación o property data binding

disabled={{expr}} ≡ [disabled] = "expr"

<p [textcont] = "expr"> ≡ {{ expr }}

<img [src] = "url">

<app [talk] = "talk"></app>

<div [ngClass] = "{selected : isSelected}">

propiedad

Angular ⟹ Modifica propiedades del arbol DOM

Los atributos NO se modifica

Esto lo hace angularjs

{{expr}}

↓

template expresi → variables / metodos ⟹ se convierte en cadena

↓

Rapida, Simple, Idempotente.

No efectos secundarios → new, asignacion ++ namespace global

Atributos que si que podemos tocar

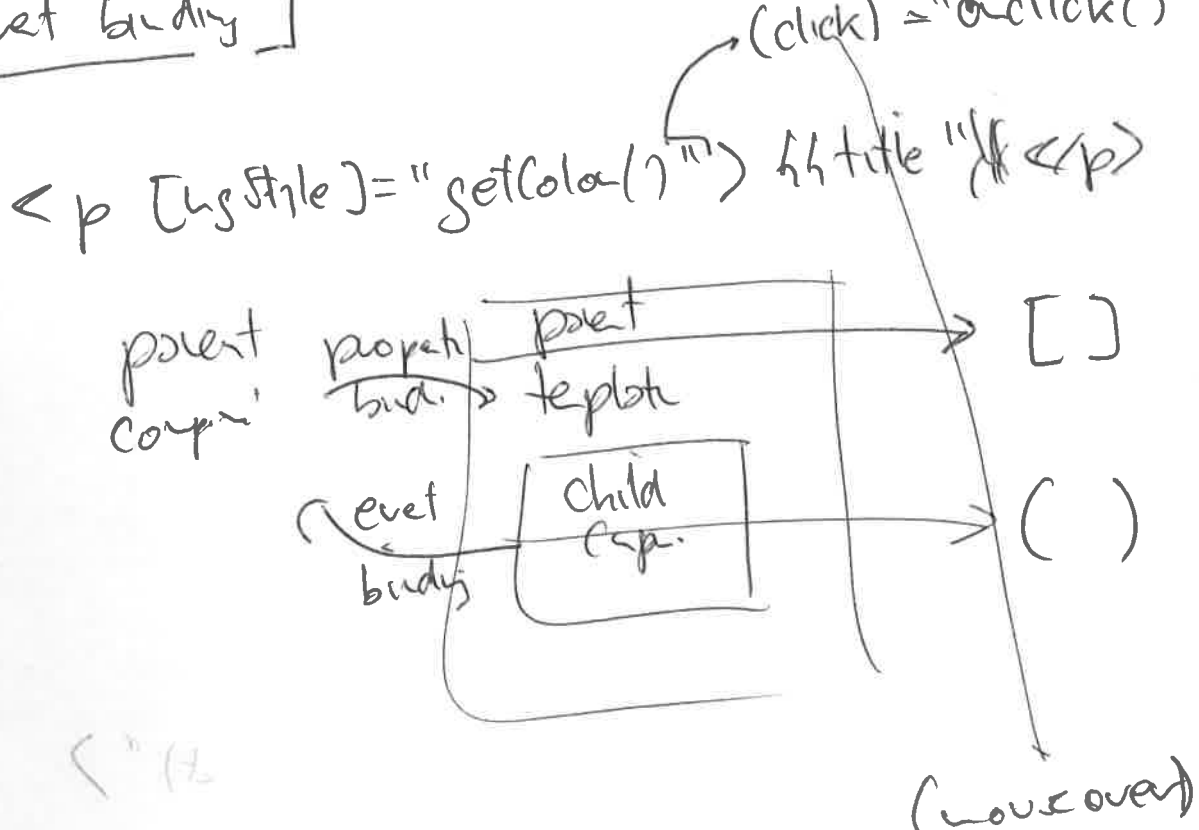
y no que no tiene DOM asociado → aria, svg, table span

[style.color] = "getColor()">

```
getColor() {
    return red;
}
```

Event binding

(1)




```
getColor() {
  return {
    'background-color' : 'red',
    'padding' : '10px'
  };
}

onclick() {
  console.log("clicked");
}
```

Pasar datos al evento

(click) = "onclick ($event)"

```
onclick($event)
{
console.log($event);
}
```

<input text (keyup) = "onclick ($event)" >

por ejemplo → $event.target.value

Necesitamos el valor

<input type = "text"

#search

(keyup) = onclick(search.value) >

local template identifier

En los expresiones de event binding si que se permite efectos secundarios

Recapitulacio de sintaxi

{{ }} → interpolacio

disabled = {{ component_variable }}

[] → property binding

[disabled] = "component_variable"

() → event binding

(click) = "component_method"

Comp ↓ template

template ↑ Component

→ declaracio variable

<input #name>

⇐ Abjo

* → structural directives

<p *ngFor = "let blk of blks">

Directivas

↳ componente sin template

⇒ Manipulo

↳ pueden aplicar varias

<... *ngFor = "let talk of talks"

↳ usa arrays

talks : string [] = ['a', 'b', 'c'];

<p *ngFor = "let talk of talks"> </p>

↓ {{talk}}

talks = string [] = [{title: 'a'},
{title: "b"}]

↑ string de objetos

{{ talk.title }}

"let talk of talks; let i = index"

{{i}} {{ talk.title }}

---

*ngIf

<span *ngIf="display(i)">{{i}}

display(i)
{ return i > 0
}

---

*ngSwitch

<div [ngSwitch]="role">

<p *ngSwitchCase="'init'">_____</p>

<p *ngSwitchDefault>_____</p>

Comunicación entre componentes

(click) = "onclick (talk)" >

}

console.log (talk)

<app-talk *ngFor = "let talk of talks" ></app-talk>

[talk(x)] = "talk"

<app-talk

Componente hijo

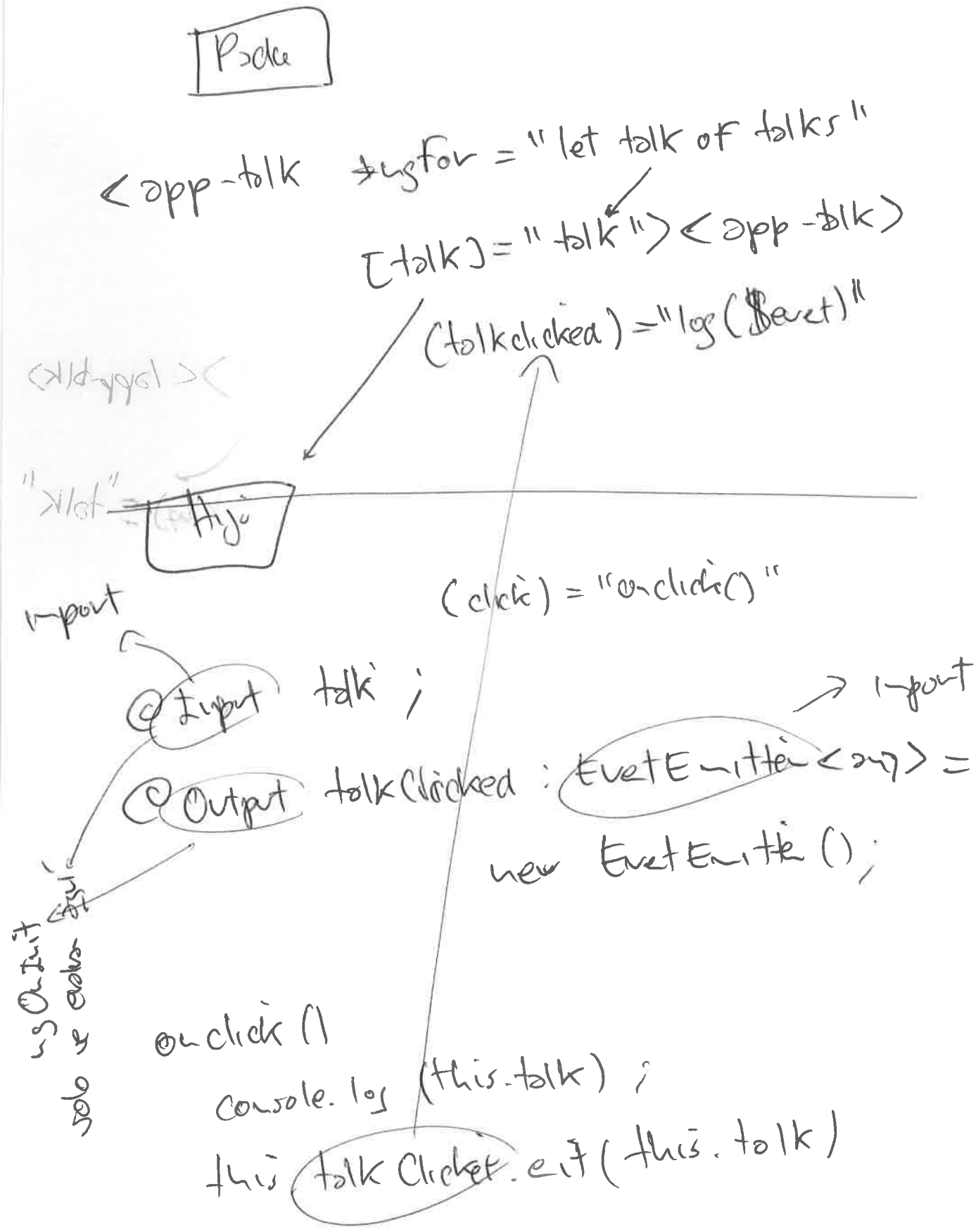
PADRE

<component [variable] = "value" />

@input variable : type

HIJO

3


Packe
<app-talk *ngFor = "let talk of talks"
[talk] = "talk"> <app-talk>
(talkClicked) = "log ($event)"
Hijo
(click) = "onClick()"
import
@Input talk ;
@Output talkClicked : EventEmitter<any> =
new EventEmitter();
import
onClick()
console.log (this.talk) ;
this.talkClicked.emit(this.talk)

Pipes

transformar de datos

{{ talk ?. title }}
{{ talk ?. fecha }}

talks : string[] = [{ title : 'hola',
fecha : new Date() }
];

Definir un pipe

{{ talk ?. fecha | date : 'HH:mm' }}

{{ talk | json }}

debug.

uppercase & lowercase

slice 1:2

number

percent

currency

Son inyectables y se puede crear nuevos.

Services

Clase con lógica de negocio
habitualmente asociada a http y Rx.

hg s s Talk

```
@Injectable
export class TalkService {
  constructor() {
    setAllTalks() {
      return ...
    }
  }
}
```

Inyección de dependencias

```
talks.component
  constructor(private talkService: TalkService)
  this.talks = this.talkService.getAllTalks();
```

Añadir a providers del módulo

debugger → pausa para el código.

TalkService es singleton

↓

solo se hace una vez.

providers: [TalkService]

↳ si es a app.module solo hay una instancia.

si queremos varios a cada módulo de componente.

Injecting the dependencies


```
constructor() {
  this.engine = new NoreEngine();
}
```

↓

```
constructor(engine: Engine)
  this.engine = engine;
```

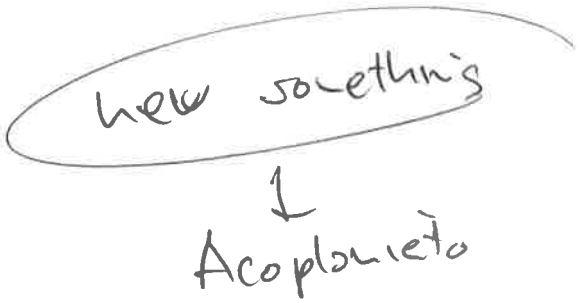
```
interface Engine {
    speed: number
}

class Car {
    constructor(private engine: Engine)
    {

    }

class SlowEngine {
    speed = 0's;
}

let exampleCar = new Car(new SlowEngine());
```

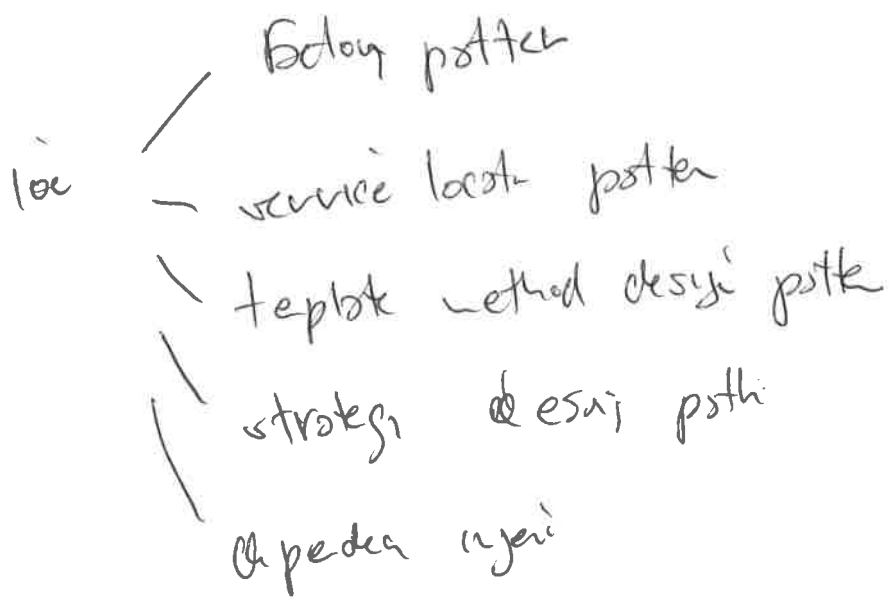


↓

Acoplamiento

Inversion of control (IoC)

|



Poke Instructi Dependen

Injection → Provide → Object

http_

↳ Vso Rx

constructor (private tokser : tok public
