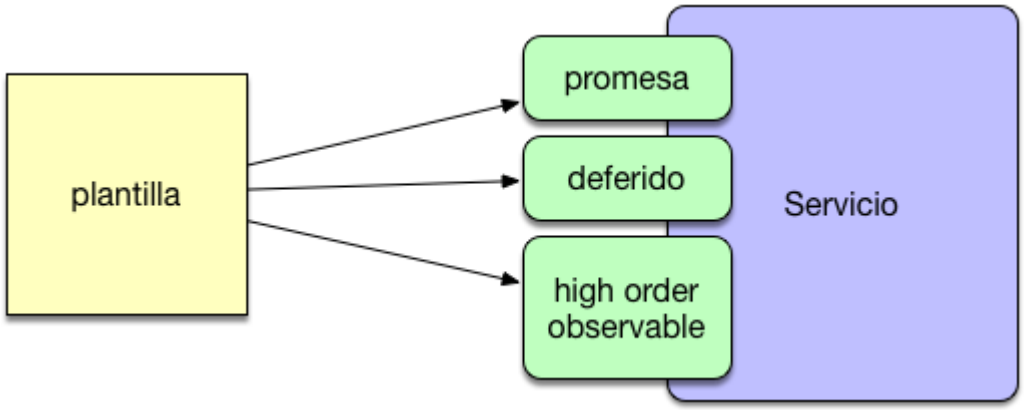
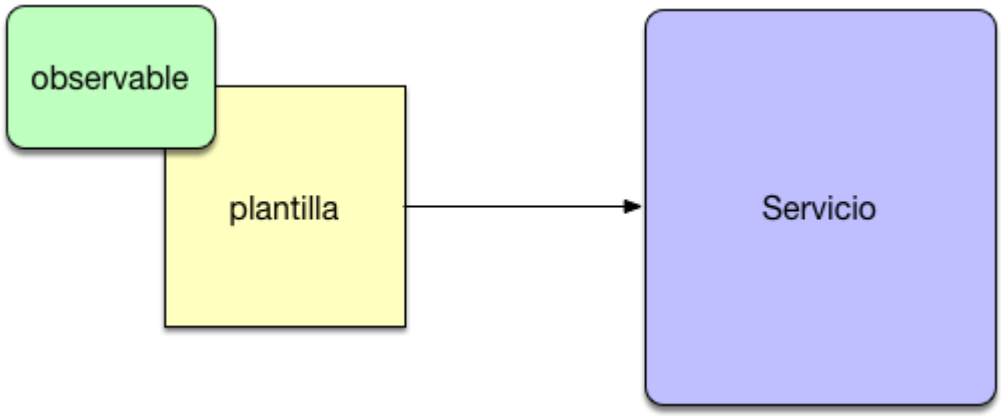
CURSO Java Herencia
GRATIS
APUNTATE!!

El concepto de Angular async pipe , es uno de los conceptos importantes a nivel de Angular.js y el manejo de la programación asíncrona. Normalmente la gestión de peticiones asíncronas siempre se ha realizado a nivel de capa de servicios utilizando promesas , objetos deferidos y high order observables.



A partir de ahora si utilizamos Angular podemos incorporar esas capacidades asíncronas directamente en la capa de presentación .



Vamos a ver un ejemplo partiendo de un componente que almacena una lista sencilla y la muestra con un ngFor.

```
<ul>

<li *ngFor="let item of lista>
  {{item}}
</li>

</ul>
```

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

@Component({
  selector: 'app-lista',
  templateUrl: './lista.component.html',
  styleUrls: ['./lista.component.css']
})
export class ListaComponent implements OnInit {
  lista:string[]=["hola","que","tal","estas"];

  constructor() { }

  ngOnInit() {
  }

}
```

Como vemos se trata de una lista sencilla que contiene unos textos. Si cargamos este componente en una vista el resultado será:



Angular Async pipe

Todo funciona perfectamente .Sin embargo existe otra opción a la hora de cargar una lista y es cargarla como una lista de observables utilizando Angular async pipe. Para ello tenemos que cambiar tanto la plantilla como el código de TypeScript para usar observables.

```
<ul>

<li *ngFor="let item of lista | async">
    {{item}}
  </li>

  </ul>
```

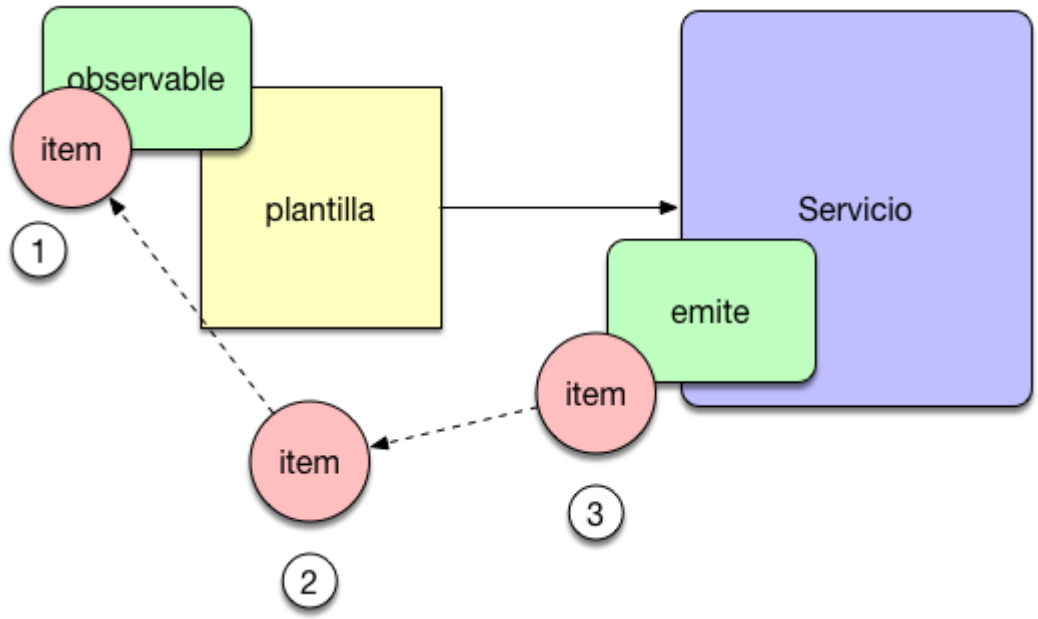
```
import { Component, OnInit } from '@angular/core';
import {Observable} from 'rxjs/Rx';
@Component({
  selector: 'app-lista2',
  templateUrl: './lista2.component.html',
  styleUrls: ['./lista2.component.css']
})
export class Lista2Component implements OnInit {
lista:Observable<Array<string>=Observable.of(["hola","que","tal","esta
s"]);
  constructor() { }
  ngOnInit() {
```

```
  }
}
```

El resultado en pantalla será idéntico:



La única diferencia es que en este caso hemos usado un observable en vez de un array normal. ¿Qué ventajas tiene esto? . Al poder asignar un observable directamente a la capa de presentación ganamos en flexibilidad ya que por ejemplo podríamos ligar un observable que emita elementos cada x tiempo.



Veamos un ejemplo similar a lo anterior .

```
import { Component, OnInit } from '@angular/core';
import {Observable} from 'rxjs/Rx';
import 'rxjs/add/operator/map'
```

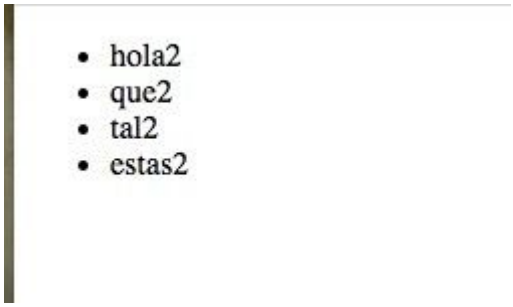
```
import 'rxjs/add/operator/zip'
@Component({
  selector: 'app-lista3',
  templateUrl: './lista3.component.html',
  styleUrls: ['./lista3.component.css']
})
export class Lista3Component implements OnInit {

lista:Observabl<string[]>=Observable.of(["hola","que","tal","estas"],
  ["hola2","que2","tal2","estas2"],["hola3","que3","tal3","estas3"]);
  listaTemporal:any;
  constructor() {
this.listaTemporal=Observable.zip(this.lista,Observable.interval(500),
(a,b)=>a)
  }
  ngOnInit() {
  }
}
```

En este caso hemos modificado la lista de textos para que sea una lista de listas y a través del operador zip de Rx.js se emita una lista cada 500 milisegundos. Al tener un Angular Async Pipe en la plantilla la lista se irá actualizando sola con las diferentes sub listas . Es decir en un primer momento aparecera:



Si esperamos medio segundo aparecera:



Si esperamos otro medio segundo aparecerá:

- hola3
- que3
- tal3
- estas3

El uso de Angular Async Pipes nos permitira integrar situaciones de programación asíncrona muy compleja en la capa de presentación como es el uso de websockets o tecnologías similares.

CURSO TYPESCRIPT
GRATIS
APUNTATE!!

Otros artículos relacionados

1. Angular ngFor la directiva y sus opciones
2. Angular 5 Hello World y su funcionamiento
3. React vs Angular 2 , frameworks vs librerias

Links externos

1. Angular y Observables