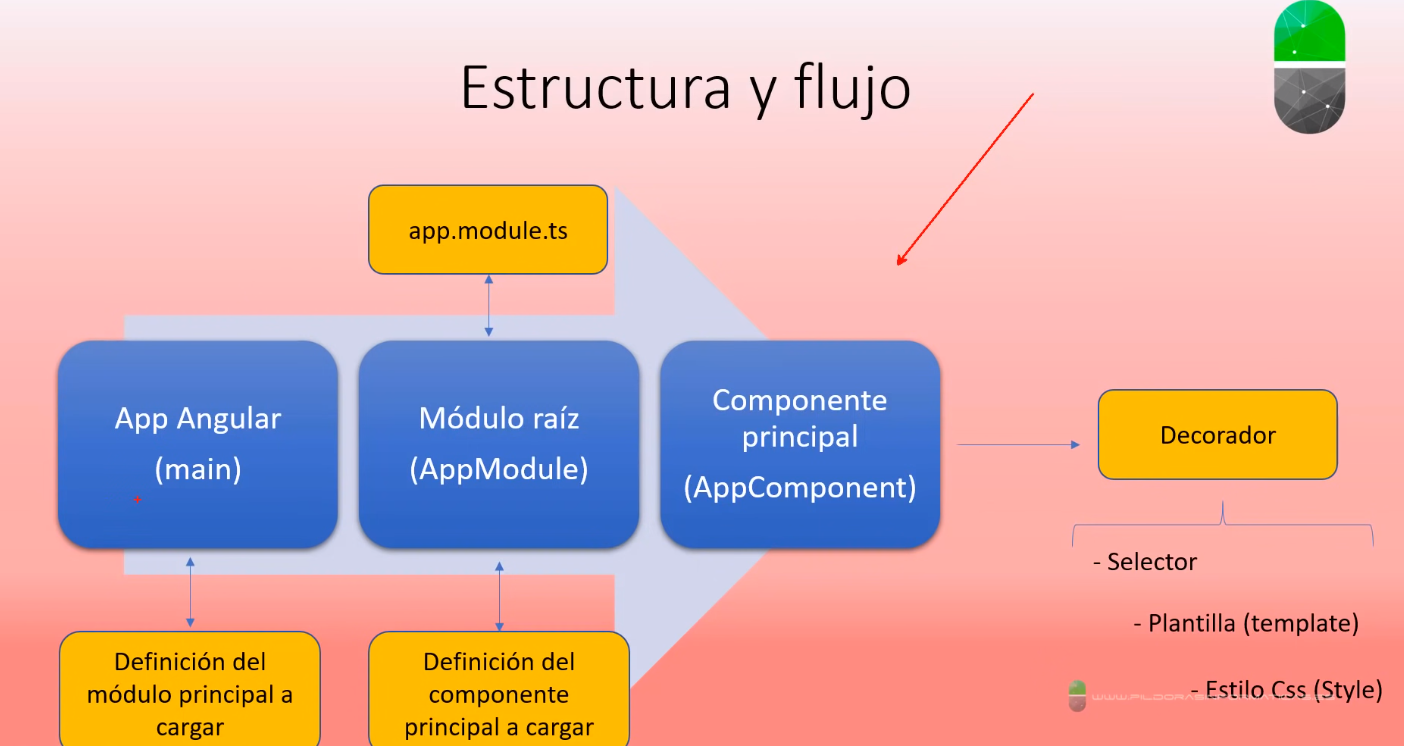
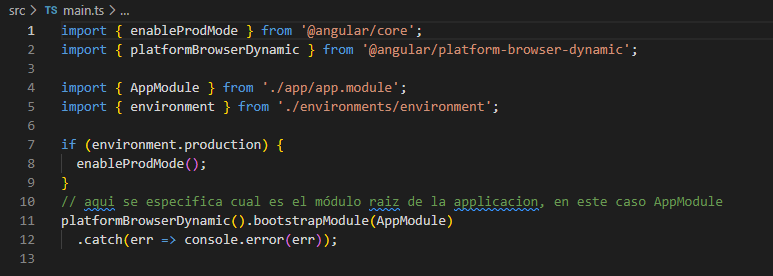
Video 4:

Estructura y flujo de una página de Angular

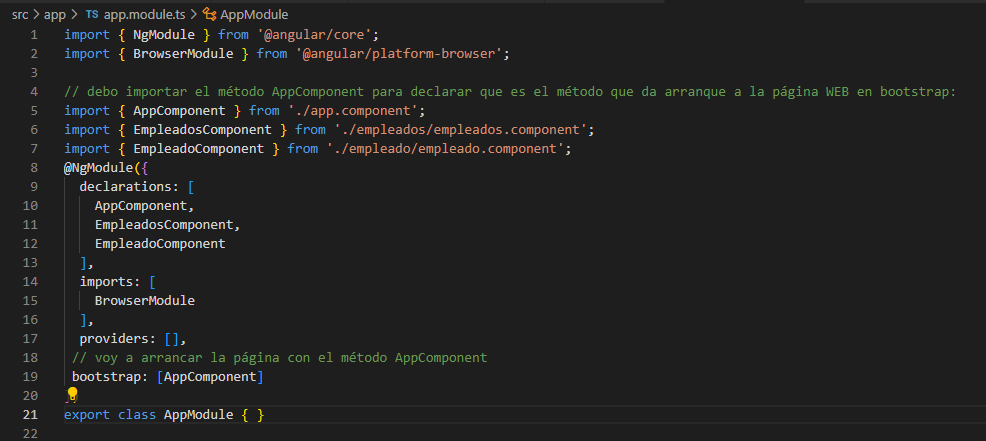


1. El punto de inicio es main. En main está la definición del modulo principal que se va a cargar, este modulo es el módulo raíz.
2. El módulo raíz está definido en el módulo app.module.ts.
3. En el módulo raíz se encuentra la definición del componente principal a cargar.
4. Por definición cuando instalamos Angular el componente principal se llama AppComponent.
5. AppComponent tiene la definición de una clase y de un decorador.

En main.ts:

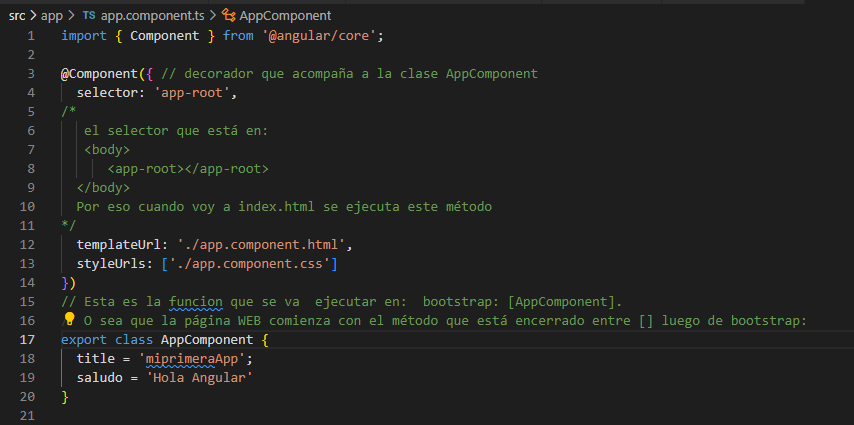


Vamos a cargar AppModule como modulo raíz:



La página va a arrancar con AppComponent porque es el componente que esta entre [ ] luego de la palabra Bootstrap:

En AppComponent vemos que tenemos el selector app-root



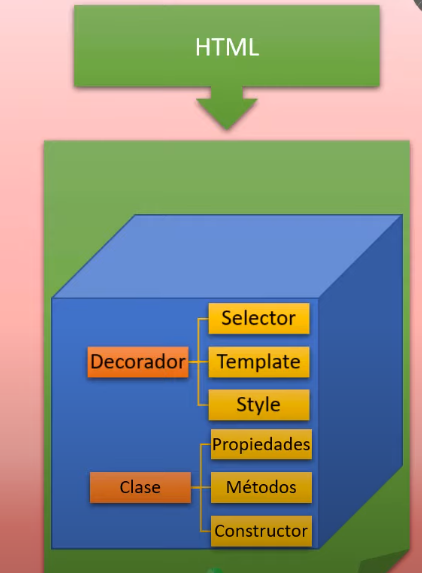
Recapitulando:

De main.ts 🡪 app.module.ts 🡪 app.componente.ts

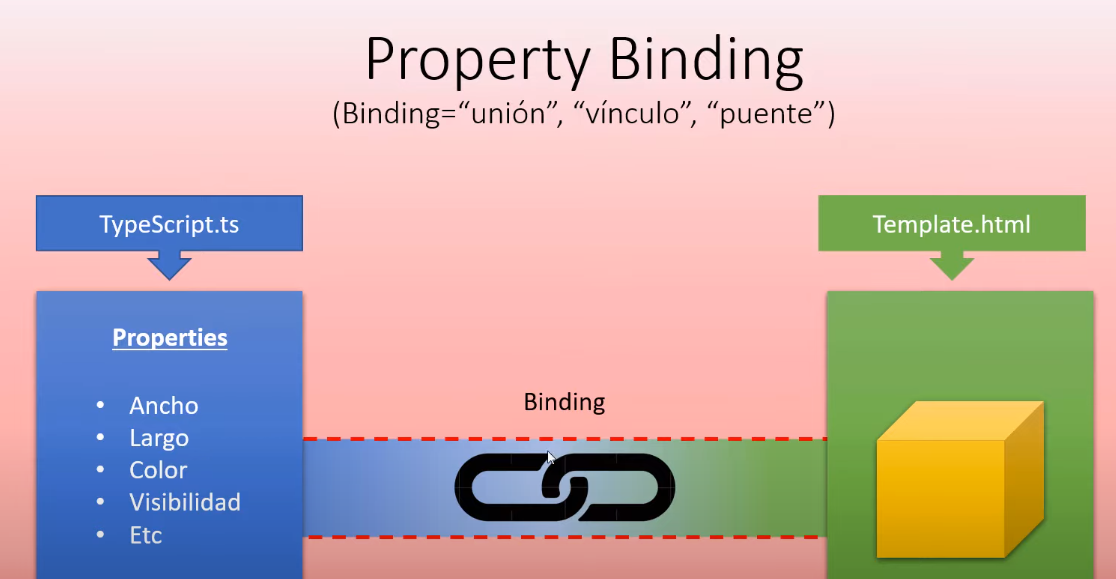
En app.component..ts especificamos:

* El selector que lo carga: ‘app-root’
* La plantilla donde se carga: ‘app.component.html’
* El style que le da formato: ‘app.component.css’

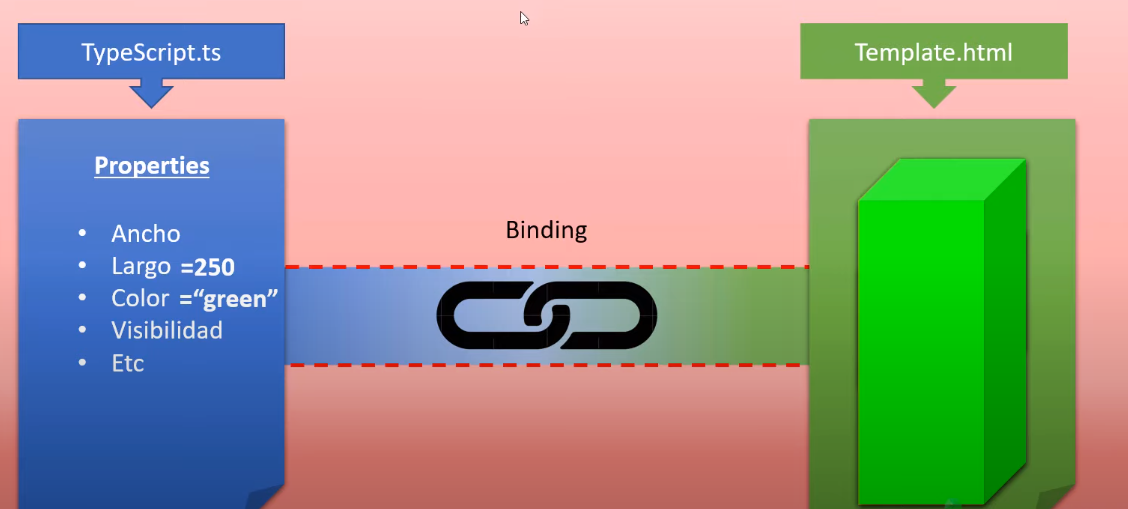
Agregar componentes



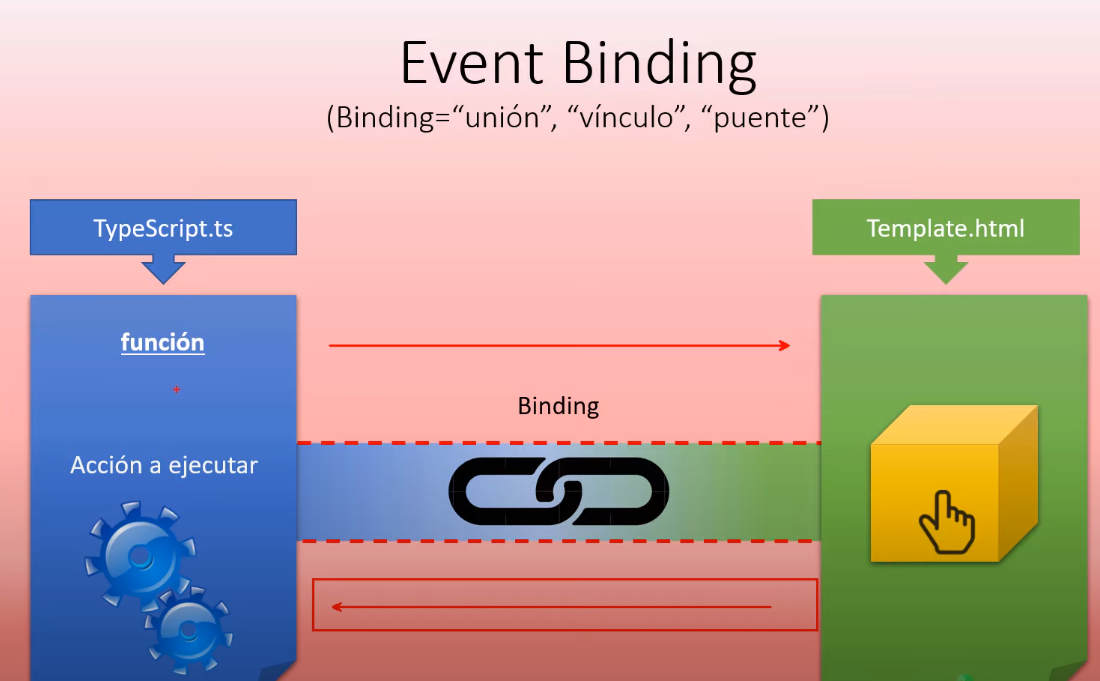
Property binding: Cambiamos las características de un objeto de forma dinámica



Cambio las propiedades

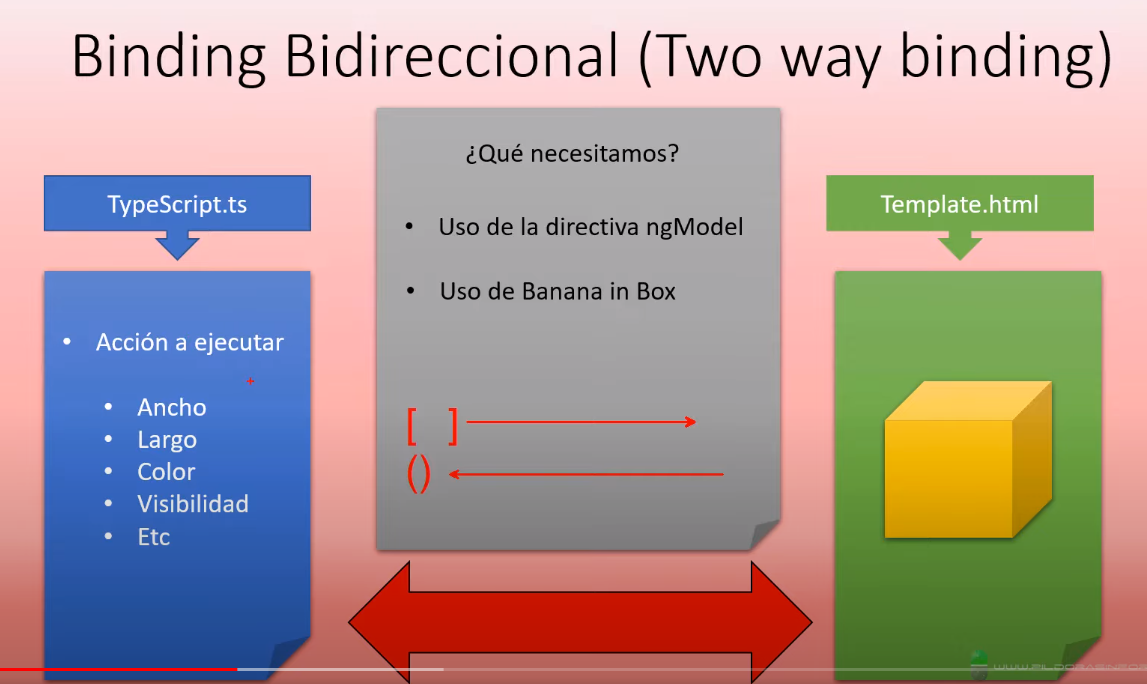


Event binding: Cuando hago una acción sobre un objeto se desencadena una función

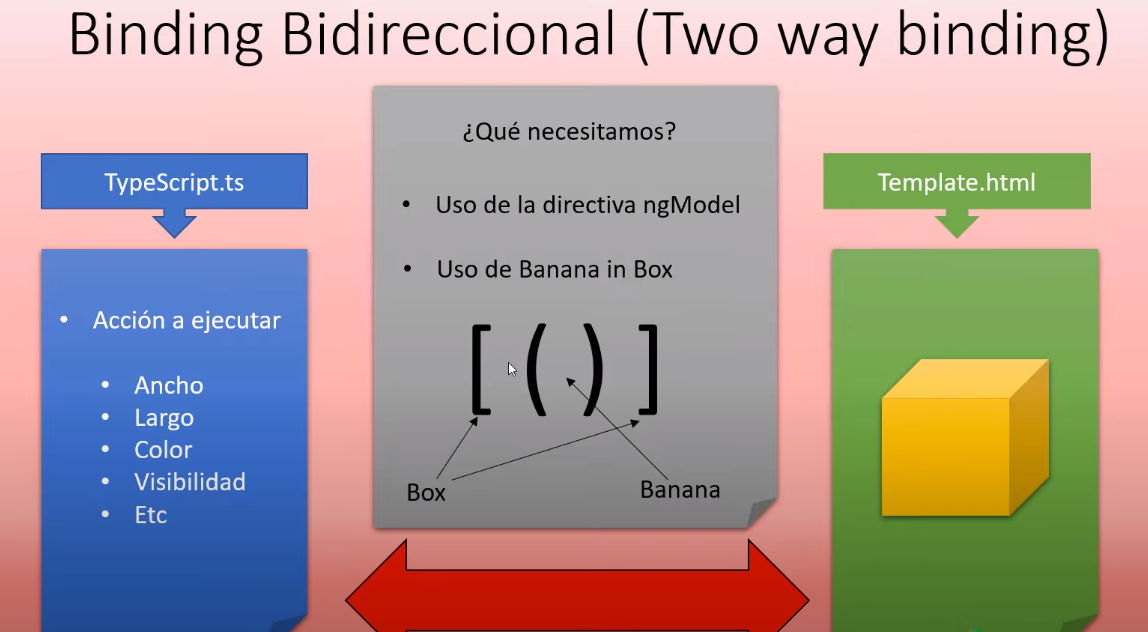


Puede ser bidireccional pero inicialmente vemos de objeto a función.

Binding bidireccional:



Property binding [ ] Event binding ( )



Two way binding: [( )] : Banana in box

No es imprescindible usar [( )] y ngModel pero facilita las cosas

**Para instalar Bootstrap:**

En la terminal escribir: npm install bootstrap –save

En el archivo angular.json agregar:

 "styles": [

              "src/styles.css",

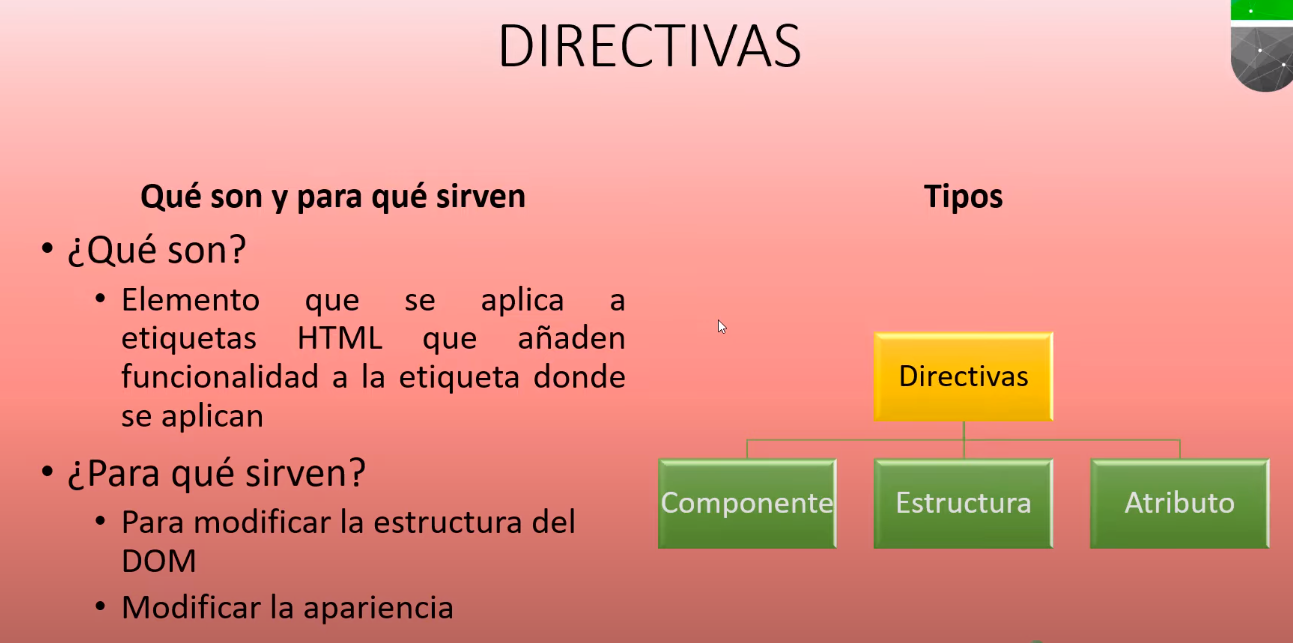
              "node\_modules/bootstrap/dist/css/bootstrap.min.css"

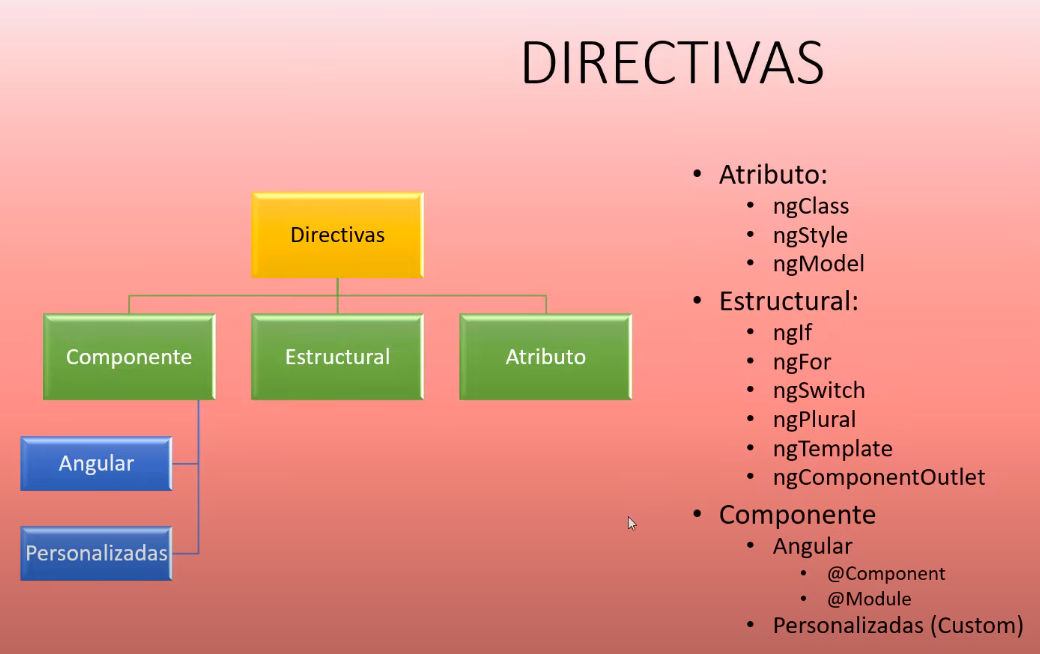
            ],

            "scripts": ["node\_modules/bootstrap/dist/js/bootstrap.min.js" ]

Ya puedo usar todas las clases de Bootstrap

**Directivas:**



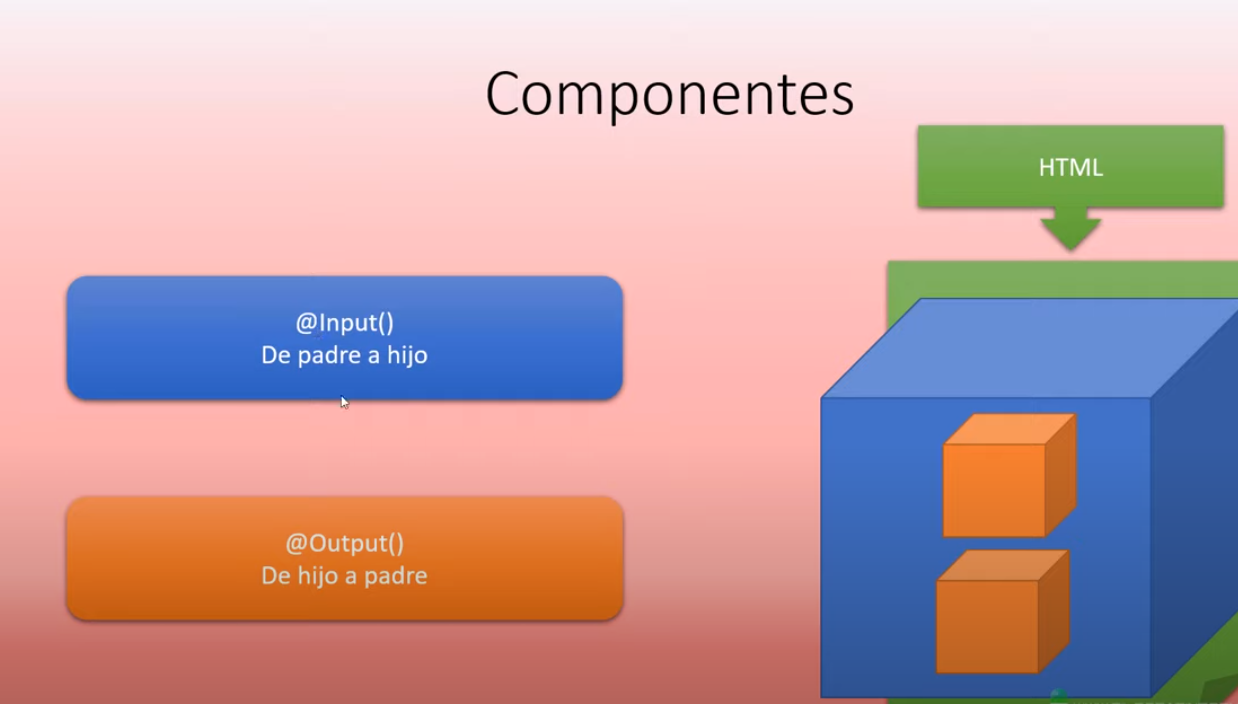


La página: trabajoDirectivaApp

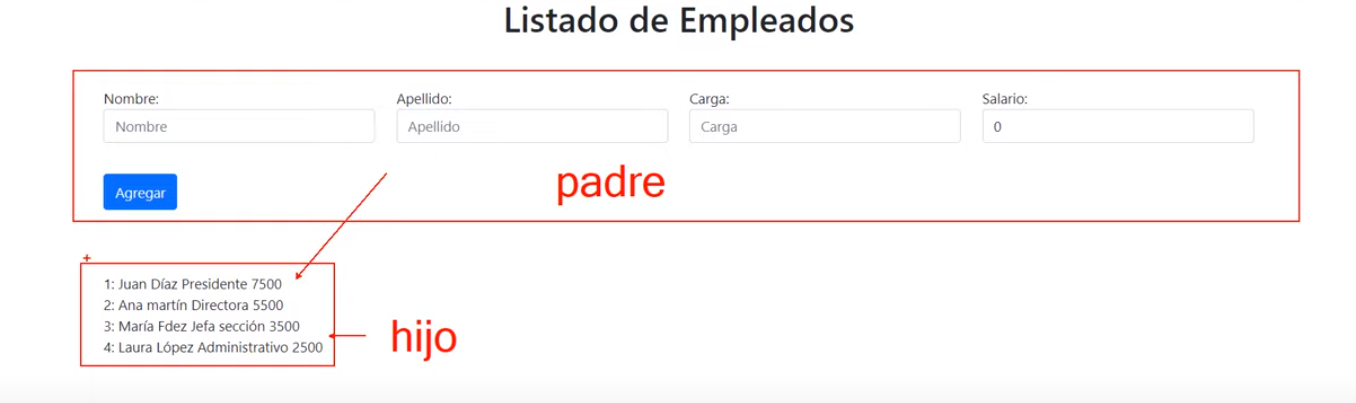
Explica:

* \*ngIf – else
* \*ngStyle y \*ngClass
* <ng-Template> asociado a else

Comunicación entere componentes:

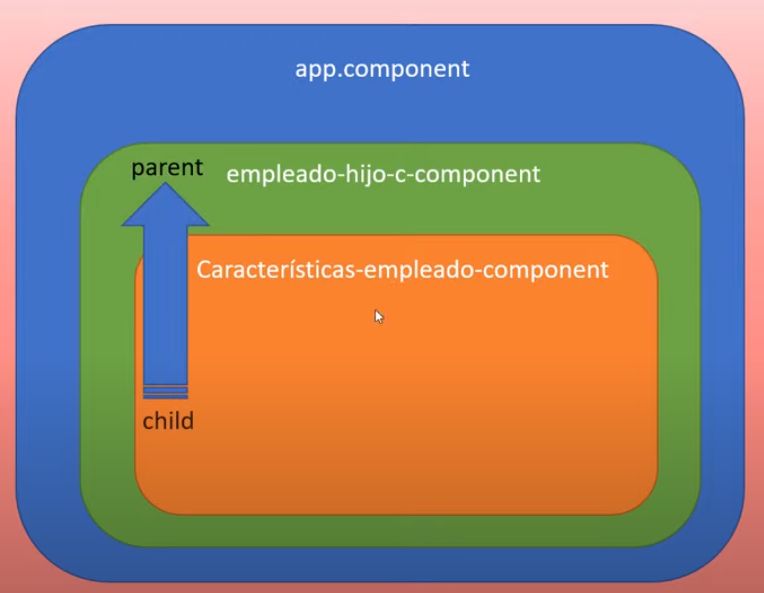


En la app-empleados quiero crear un componente hijo que muestre la lista de empleados:



El formulario estará en el componente padre y el listado en el hijo.

Ahora hacemos el flujo inverso:

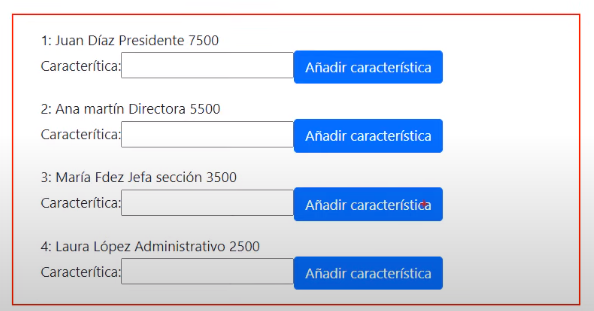


Vamos a crear un mini formulario con datos adicionales para pasar del hijo al padre

Componente principal:



Este es el componente hijo:



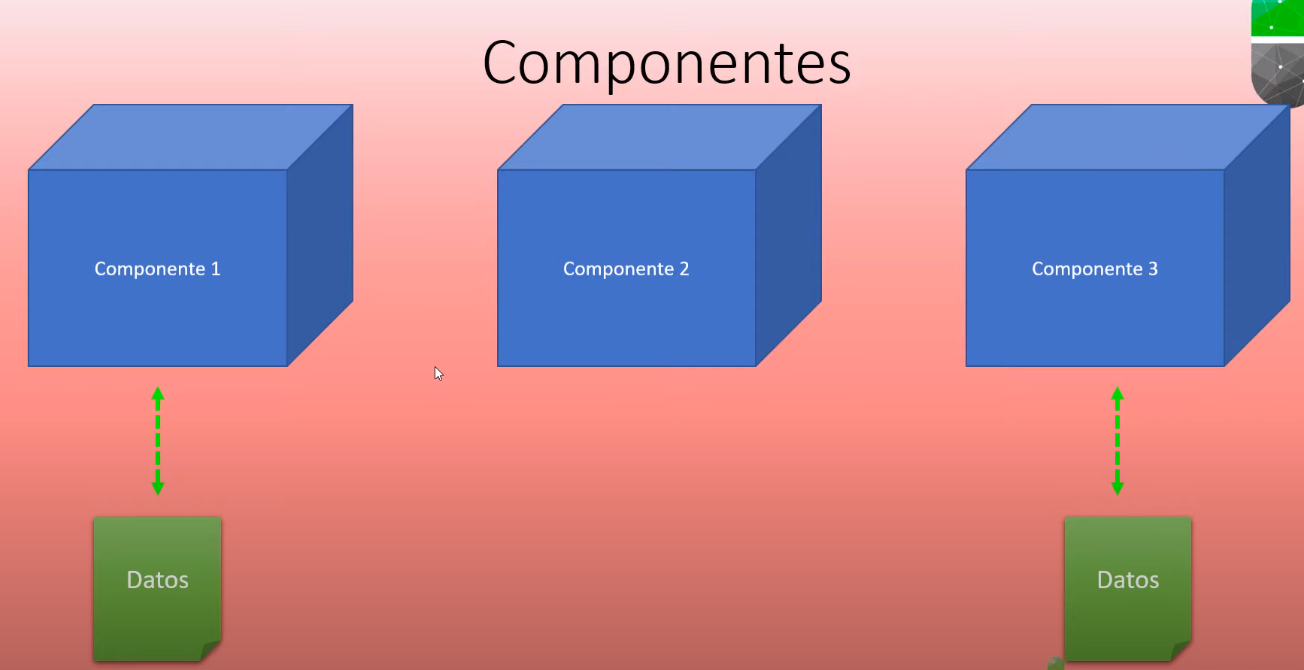
Este es el componente nieto:



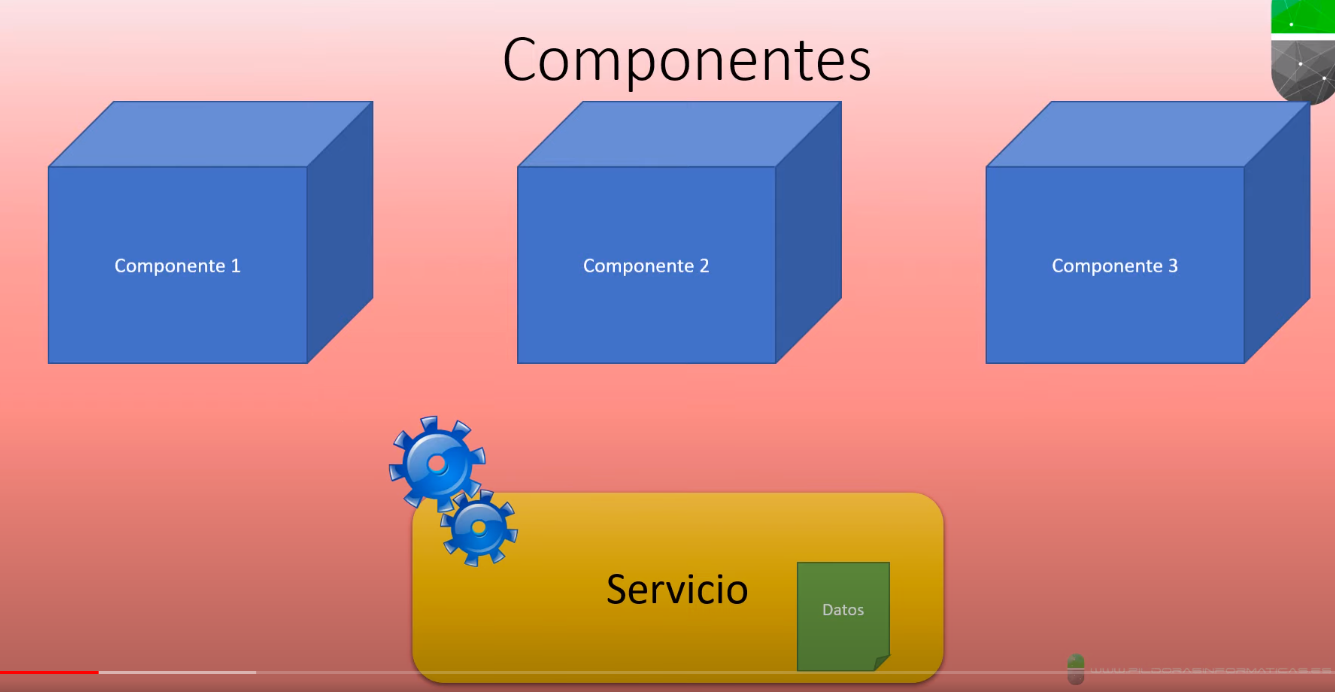
Que es el que conectamos al padre al cual hemos llamado anteriormente componente hijo

Las características que ingresamos en el componente hijo pasan al listado del componente padre.

Servicios:

Supongamos que dos componentes tienen que extraer datos de una base de datos

Para evitar repetir el código en cada componente creamos un servicio

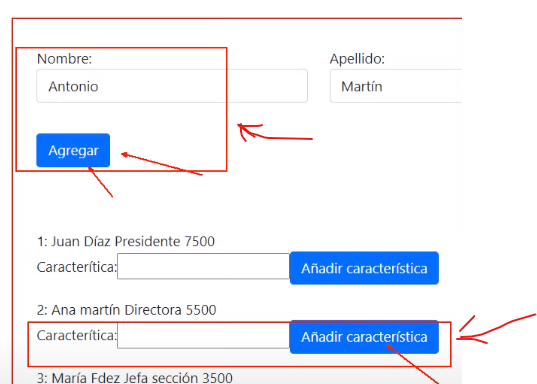


Se dice que es una inyección de dependencias porque cuando un componente necesita un dato el servicio se lo inyecta.

En el proyecto de empleado quiero que antes de agregar la característica del empleado la misma me aparezca en una ventana emergente.

El servicio es que salga la ventana emergente con la información

A estos dos componentes le vamos a agregar este servicio



`

Vamos a poner los datos de los empleado en un servicio por si necesitamos compartirlos con otros componentes

**Routing**

Queremos hacer un menú con diferentes opciones. Cada opción va a ser un componente diferente, tenemos que escribir cada uno de los componentes. Vamos a pasar lo que teníamos en app.componente a un componente home

