agregar un componente (o una página) en un módulo Angular

GH Gustavo Hernán Dohara

Ahora nos metemos en la creación de un componente, o sea, una página HTML, la página que vemos cuando abrimos nuestro browser.

Entonces… ¿qué es un Component ?

Conceptualmente hablando y de manera muy simple, un Component es HTML con su lógica. Pudiendo ser ese HTML una página completa o solo una sección de una página, por eso un componente puede estar formado por más componentes dentro de él.

Los Componentes están compuestos por tres grandes elementos:

Template /Class / Matadata

- **Template :**

  Es el HTML o la capa de vista (o View Layout). Es lo que el usuario ve en el browser, imágenes, título, texto, vídeos, etc.

- **Clase o Component :**

  Es la lógica del Template. Es una Clase en TypeSript (al igual que los módulos Angular) y es a lo que comúnmente se llama el Componente. Esta clase contiene las propiedades (o datos) que estarán disponibles en la vista, además de contener la lógica necesaria para la Vista.

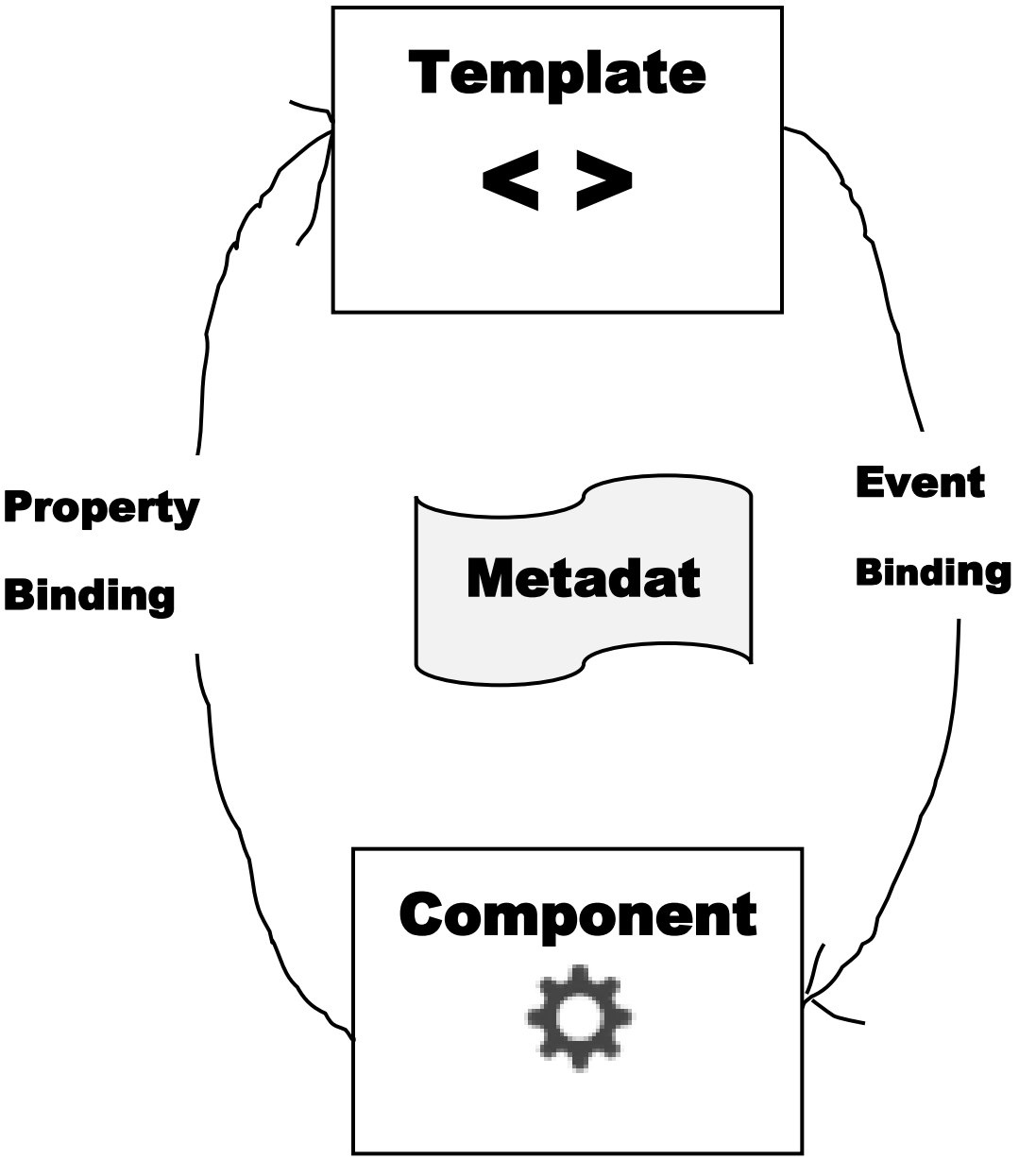
- **Metadata :**

  Es un Decorador, los Componentes se decoran con «@Component», (al igual que los módulos Angular, que se decoraban con «@ngModule«). Son datos extra necesarios para que Angular Binde (o «una») el Template con el Component. Además del decorador «@Component» existen otros más que decoran los atributos de la Clase y los argumentos de los métodos.

Si bien se pueden diagramar los tres elementos por separado (como se muestra en la imagen mas abajo), conceptualmente se puede ver al «Componente» como el conjunto de la

- **Clase**
- **Template**
- **su Metadata**

como si fueran un único elemento


Template
< >
Property
Binding
Metadat
Event
Binding
Component

Cabe aclarar que un Componente tiene SÓLO UN Template; a diferencia de AngularJS (o Angular 1), en donde una página podía tener varios Controladores (el equivalente a los componentes en Angular)

Si seguiste los pasos para crear un proyecto en Angular usando angular-cli sabrás que, por defecto se te crea un componente llamado «App« de hecho te crea la

Clase -> app.component.ts

Template -> app.component.html

hoja de estilos -> app.component.css

archivo test -> app.component.specs.ts

- `app.comonent.css`:
  Es la hoja de estilos, en este archivo se agregan colores, márgenes, tamaño de texto, y todo lo referente a cómo se ven los componentes del archivo HTML (el Template). Este archivo viene por defecto cuando creamos un proyecto con angular-cli por si lo llegamos a necesitar ;)

- `app.component.html`:
  Es el Template, o sea el HTML, la página que se ve en el browser. Si levantamos nuestra aplicación (con el comando `ng serve`) y la abrimos en nuestro browser, podemos ver que el browser entiende nuestro código HTML y lo convierte a lo que finalmente vemos en la pantalla.

- **`app.component.spec.ts`: Es un archivo para Test Unitarios, o sea para testear sólo este Componente, te lo crea angular-cli con un test muy simple para que vos vayas completando con los test que necesites para tu Componente (es muy importante testear así que no te olvides de completarlo ;)).**

- `app.componnet.ts`:

  Esta es la Clase TypeScript, dentro esta la lógica del Componente y la Metadata.

  Este es su contenido:

```
import {Component}from'@agular/core';

@Component ({

      Selector:'app-root',
   templateUrl:'./app.component.html',

    styleUrls: ['./app.component.css']

   )}

export class AppComponent {

      title = 'app';

   }
```

// Explicación

```
@Component
```

//Este es el METADATA :El decorador

```
Selector:'app-root',
```

// Con este selector se puede incluir un.

// Componente dentro de otro. Es el

// identificador de este Componente

```
templateUrl:'./app.component.html',
// Indica la ubicación del Template
// asociado a este componente y solo
// puede haber uno.

styleUrls: ['./app.component.css']
// Hojas de estilo. Puede haber
// muchas,a diferencia de templateUrl
```

```
export class AppComponent {

// la Clase Typescript (el Componente)

    title = 'app';

// Una propiedad (o dato) de la clase
// que es visible en el template
```

Este es la **Metadata** (el decorador **@Component**)

Es el identificador de este **Componente**. Con este selector se puede incluir un Componente dentro de otro

templateUrl indica la ubicación del **Template** asociado a este Component (fíjense que solo puede ser uno solo)

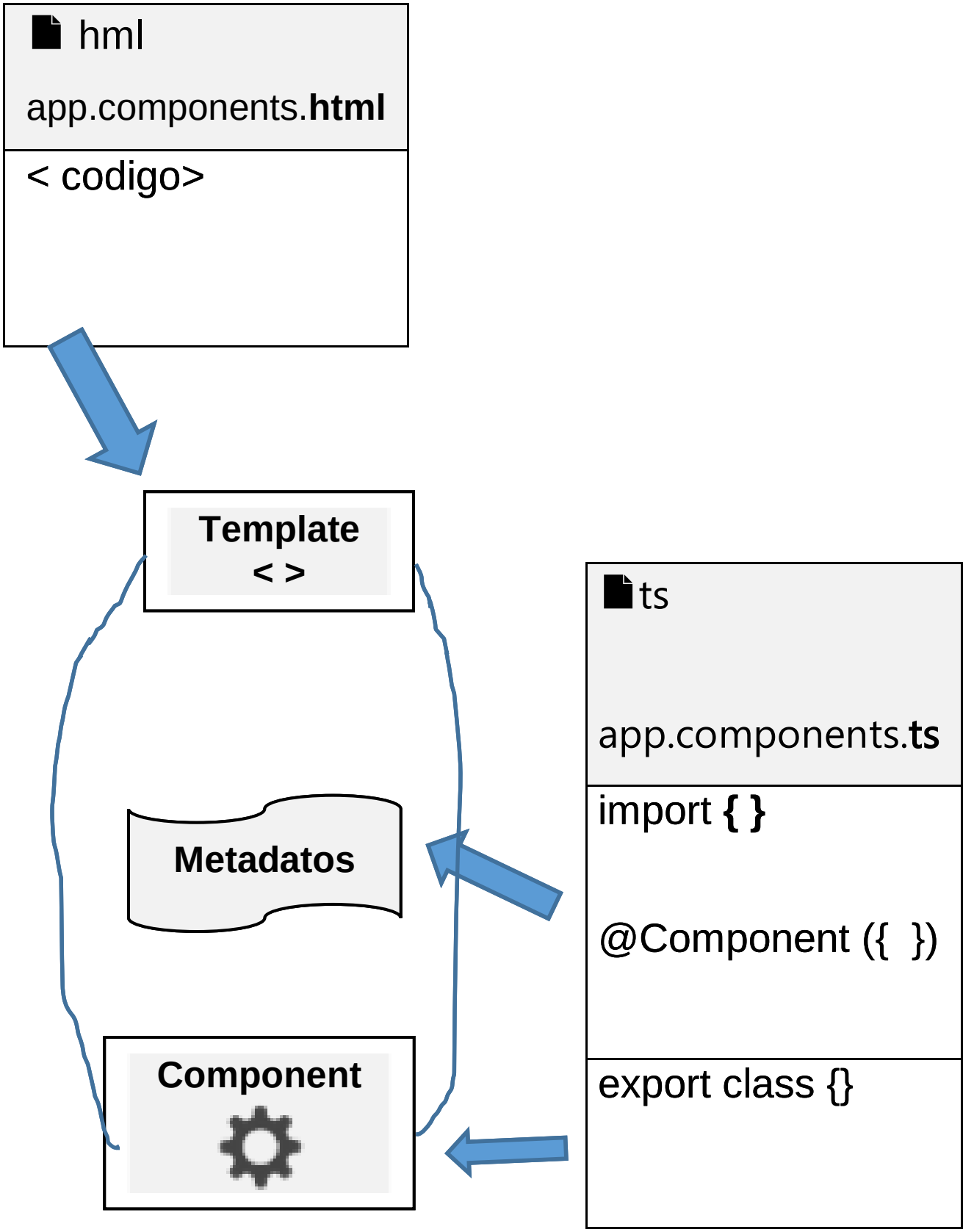
Es la lista de **hojas de estilo** (fíjense que acá se pueden ser muchas, a diferencia del **templateUrl**)

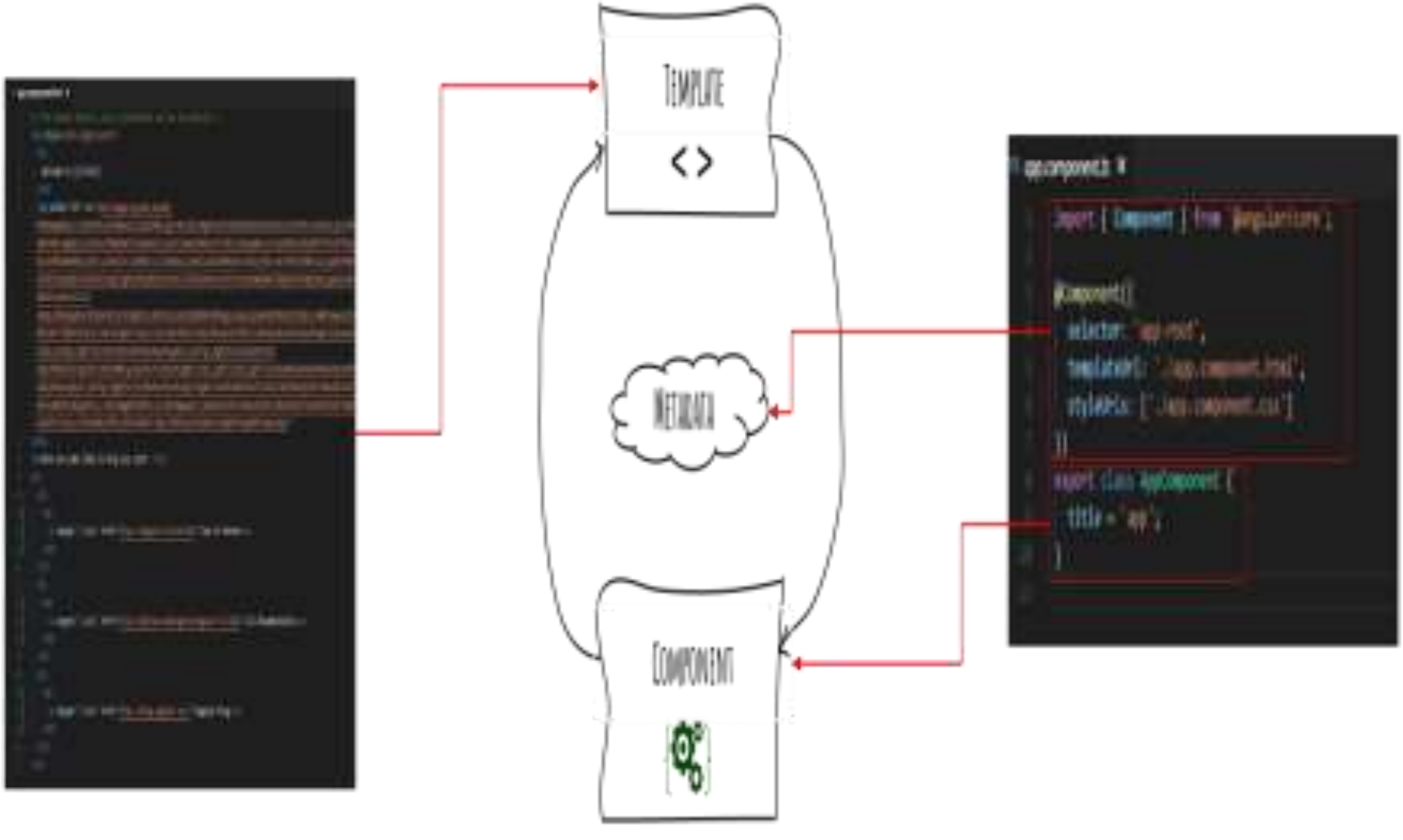
La clase **Typescript** (el **componente**)

Una **propiedad** (o **dato**) de la clase que es visible en el **Template**

Cuánta información para tan pocas lineas de código, ¿no? Bueno, con el tiempo te vas a acostumbrar porque todos los Componentes se construyen de forma similar.

Volviendo al modelo conceptual de más arriba, ahora lo podemos relacionar con el código que vimos, de esta forma, y cerrar un poco mas la idea:


hml
app.components.html
< codigo>
Template
< >
Metadatos
Component
ts
app.components.ts
import { }
@Component ({ })
export class {}

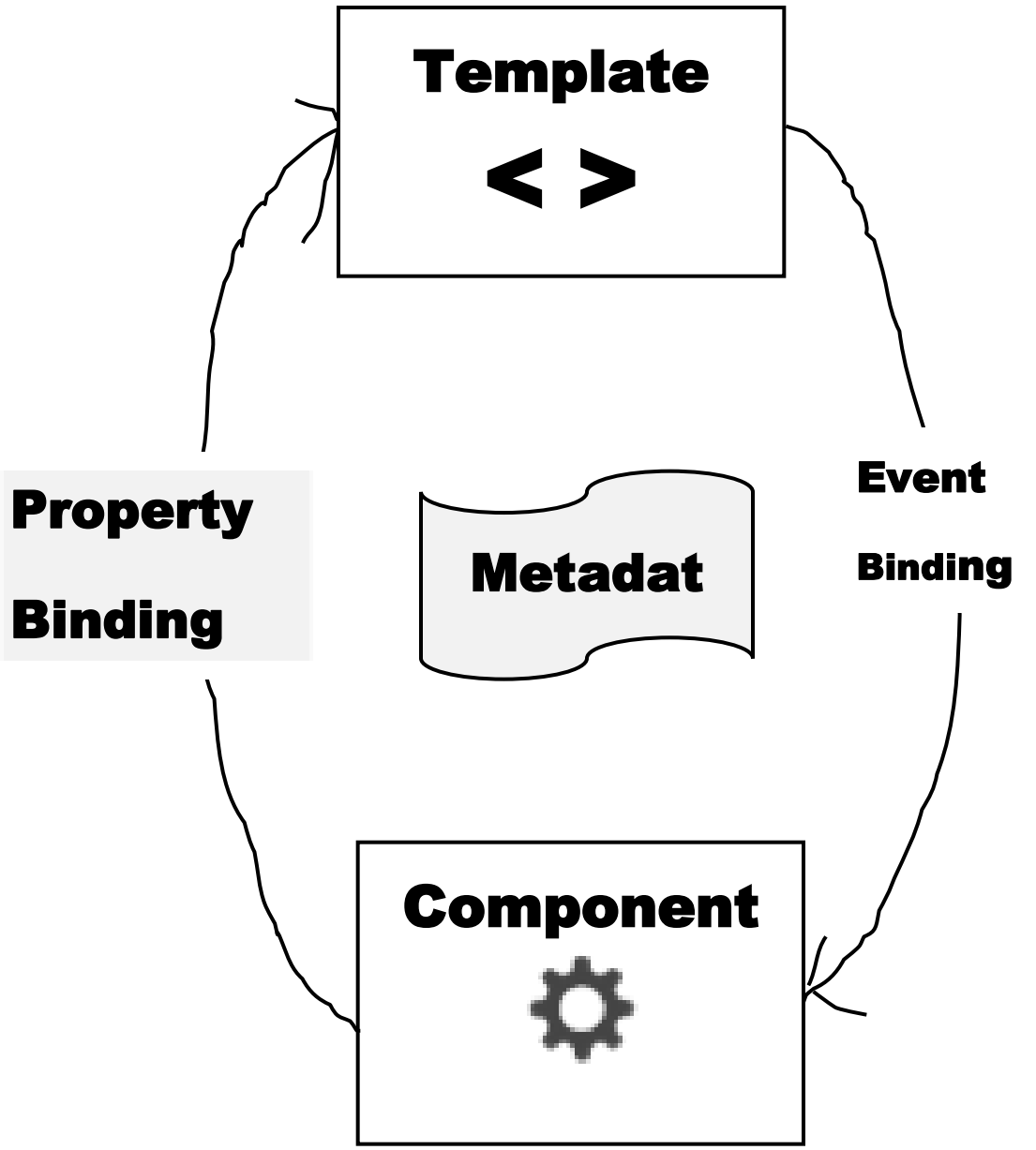

TEMPLATE
<>
METADATA
COMPONENT

Para finalizar, nos falta explicar cómo es que aparece la palabra «app» (el nombre del componente en el browser. Si no te diste cuenta, hay algo raro (o mágico) en el browser, y es la palabra «app». En el browser aparece peeeeero si te fijás, en el HTML no aparece esa palabra por ningún lado. ¿Aparece mágicamente?

ESTE ES EL TÍTULO, PERO
¿¿DE DONDE SALE LA
PALABRA "APP"??

Property Binding:

Es el mecanismo de «unir» (o binding) un dato entre el Template y el Component (la clase). Este mecanismo se encarga de refrescar automáticamente el dato bindeado en el browser: cada vez que cambiemos el dato en la Clase, el cambio se verá refrescado en el browser sin que hagamos nada,


Template
< >
Property
Binding
Metadat
Event
Binding
Component

Para que el Template sepa que un dato está bindeado, se lo encierra entre doble corchetes

{{ }}

Es la forma de decirle al Template: «¡che Template! fijáte que el dato entre corchetes dobles es un dato bindeado en la Clase.

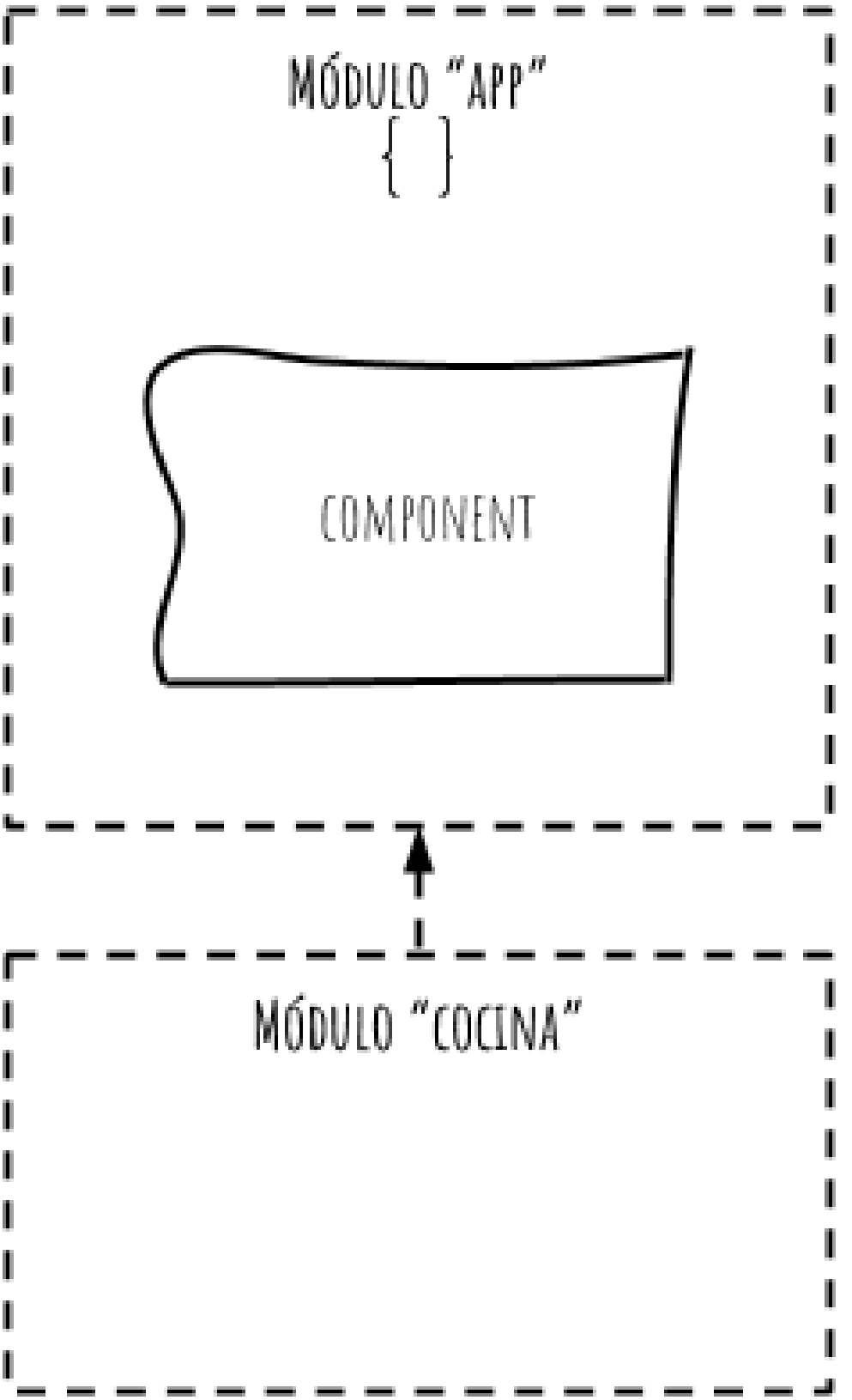
hml app.components.**html**
  Welcome to **{{title}}**

<<app>>

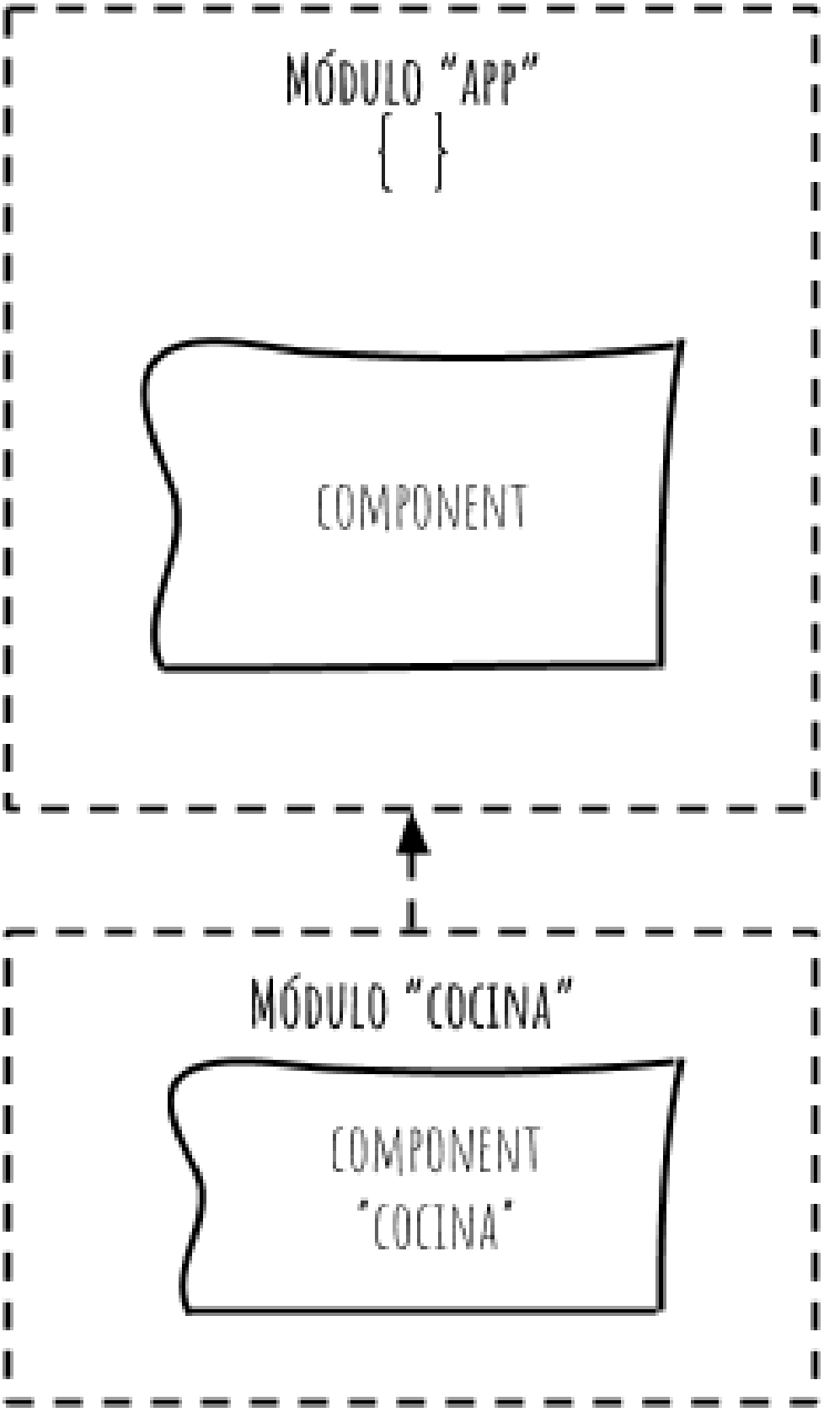
Volviendo al ejemplo de nuestra «Casa Inteligente» tenemos

un modulo llamado «app»;

en este modulo app se importa el módulo llamado «cocina»


MÓDULO "APP"
{ }
COMPONENT
MÓDULO "COCINA"

Lo que vamos a hacer es agregarle al módulo cocina un Componente llamado «cocina» (todo muy original) usando nuestra querida herramienta angular-cli.


MÓDULO "APP"
{ }
COMPONENT
MÓDULO "COCINA"
COMPONENT
"COCINA"

// Creamos un proyecto de cero llamado
// mi-casa-inteligente

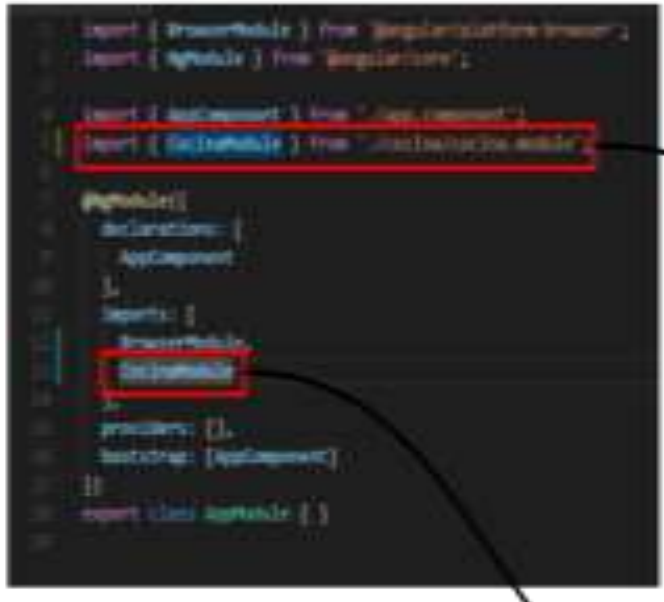
```
ng new mi-casa-inteligente
```

```
cd mi-casa-inteligente
```

// creamos el módulo «cocina»

```
ng generate module cocina
```

// Agregamos el módulo cocina en
// app.module.ts



Me importo el módulo
CocinaModule
(de la forma TypeScript)

Módulo "app"
{ }

component

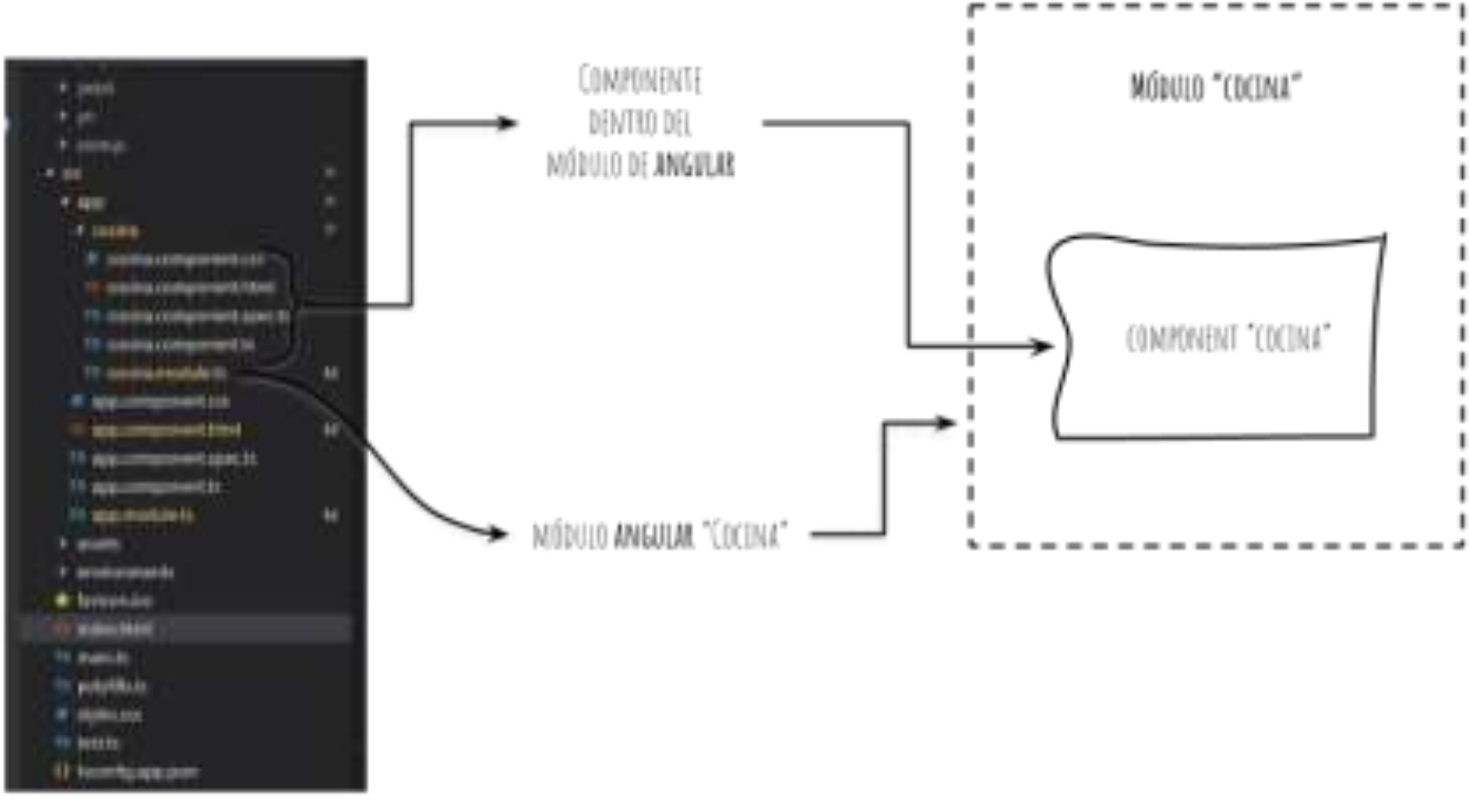
El Módulo "App"
usa el módulo
"Cocina"

Módulo "cocina"

```
ng generate module cocina
```

/* Lo que hacemos es crear («generate«) un componente («component») llamado «cocina» en el módulo (–module) llamado «cocina».

Este comando nos crea los archivos del componente «cocina» y ademas nos agrega AUTOMÁTICAMENTE el componente en el módulo. ¡Nos deja el componente listo para usar! */


COMPONENTE
DENTRO DEL
MÓDULO DE ANGULAR
MÓDULO "COCINA"
COMPONENT "COCINA"
MÓDULO ANGULAR "COCINA"

Como mencionamos antes, se crea el **Template** (cocina.component.html), la **hoja de estilos** (cocina.component.css), el **archivo de test unitario** (cocina.component.spec.ts) y la **clase** (cocina.component.ts).

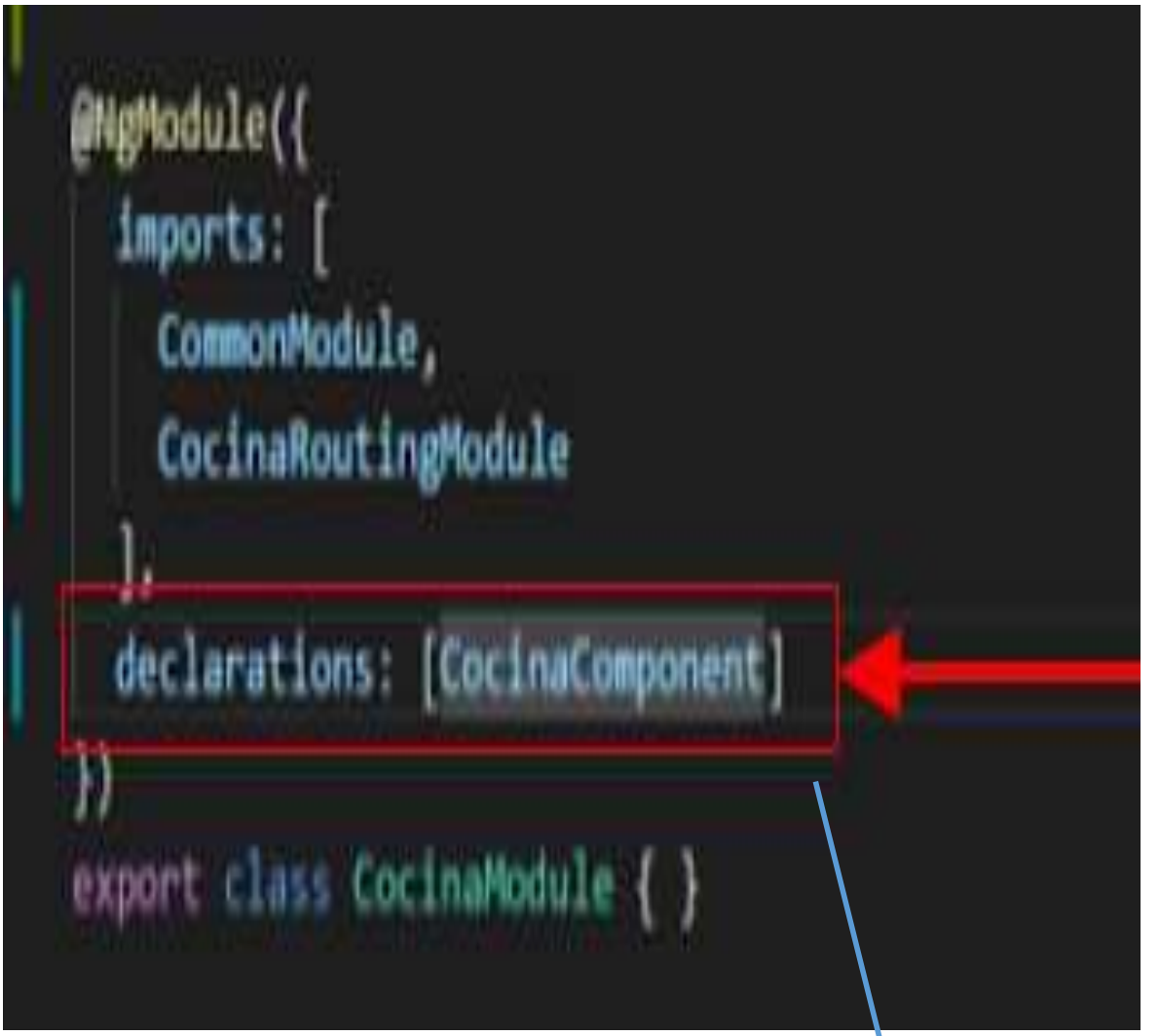
Por otro lado, también se modificó el archivo cocina.module.ts, indicándole al componente (CocinaComponent) que se incluya en el módulo «Cocina»

Es muy importante no confundir un Componente con un Módulo Angular.

Si ves bien,

el Componente lo estamos agregando en el array de «declarations»

y no lo estamos Importando (agregando en el array de imports, como es el caso de los otros Módulos Angular)



La forma que tenemos a **Angular** de decirle que este componente se creo en este Módulo **Angular**

¡Conclusión!

La creación de componentes es una de las tareas que más vas a tener que hacer, es por eso que mejor te aprendas bien la forma de hacerlo.