

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Tema: Angular

- ¿Qué es Angular?
- Instalación de Angular y software necesario
- Creación de primera aplicación
- Estructura de aplicación Angular
- Creación de componentes
- Binding

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Angular (Introducción)



# Módulo 4 - Front End - Desarrollo Web Dinámico

## ¿Qué es Angular?

- **Angular** es un framework para crear aplicaciones web SPA (Single Page Aplicación).
- La particularidad es que permite generar una aplicación web de una única página existen otros framework que permite generar aplicaciones web pero con varias páginas.
- Esto se da gracias a la tecnología de componentes con la que trabaja este framework.
- Fue creado por Google



# Módulo 4 - Front End - Desarrollo Web Dinámico

## ¿Qué características presenta Angular?

- Separa **frontend** y **backend** de la aplicación.
- Simplifica el código.
- Sigue el patrón **MVC**.
- Basado en componentes.
- Es de código abierto.
- Utiliza como lenguaje de programación a **TypeScript**.



# Módulo 4 - Front End - Desarrollo Web Dinámico

## ¿Por qué utilizar Angular?

- Se desarrollo de aplicaciones es rápida al igual que la navegación que se logra en ella.
- Es modular lo que nos permite la reutilización de códigos.
- Fácil mantenimiento por utilizar tecnología MVC y Componentes.
- Es multiplataforma.
- Futuro estable.
- Gran soporte de herramientas y una comunidad muy activa.
- Creciente demanda.



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Instalación, Configuración y desarrollo de un ejemplo en Angular

(Instalación en windows 10)

# Módulo 4 - Front End - Desarrollo Web Dinámico



**Instalando NODEJS y  
NPM (Gestor de paquete de NODE)**



# Módulo 4 - Front End - Desarrollo Web Dinámico

**Ver proceso de instalación de la PPT Modulo 3 TypeScript**

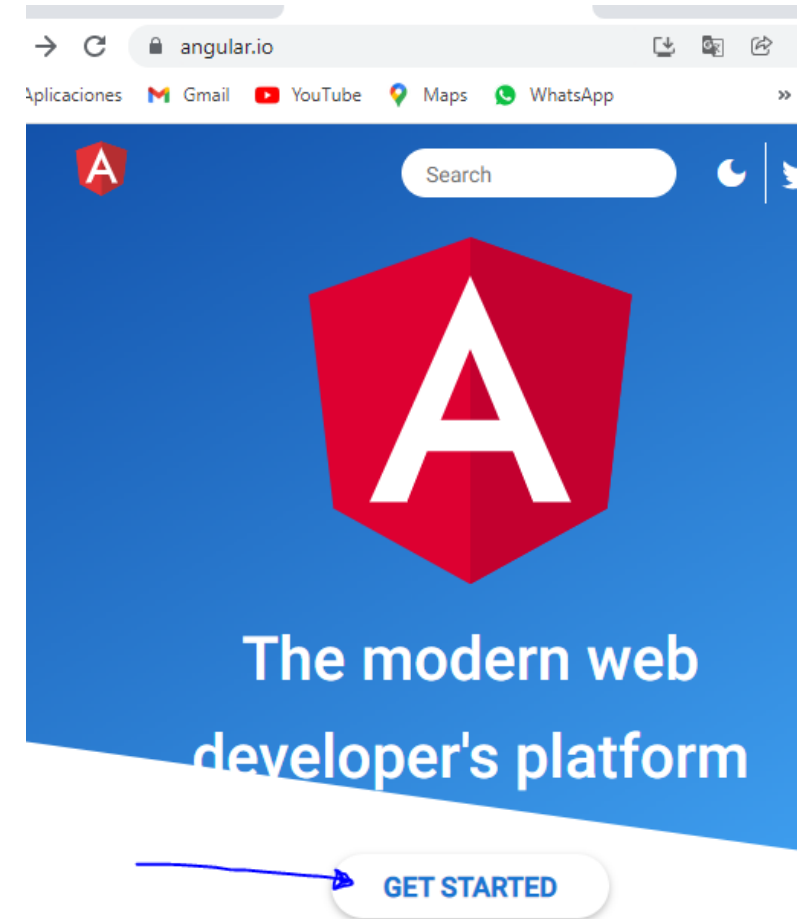


# Módulo 4 - Front End - Desarrollo Web Dinámico

## ¿Instalación de Angular?

Accediendo a su sitio oficial contamos con el proceso de instalación de todas las aplicaciones y demás paquetes para poder trabajar con Angular.

<https://angular.io/>



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Angular CLI (Command Line Interface) (Instalación en VSCode)

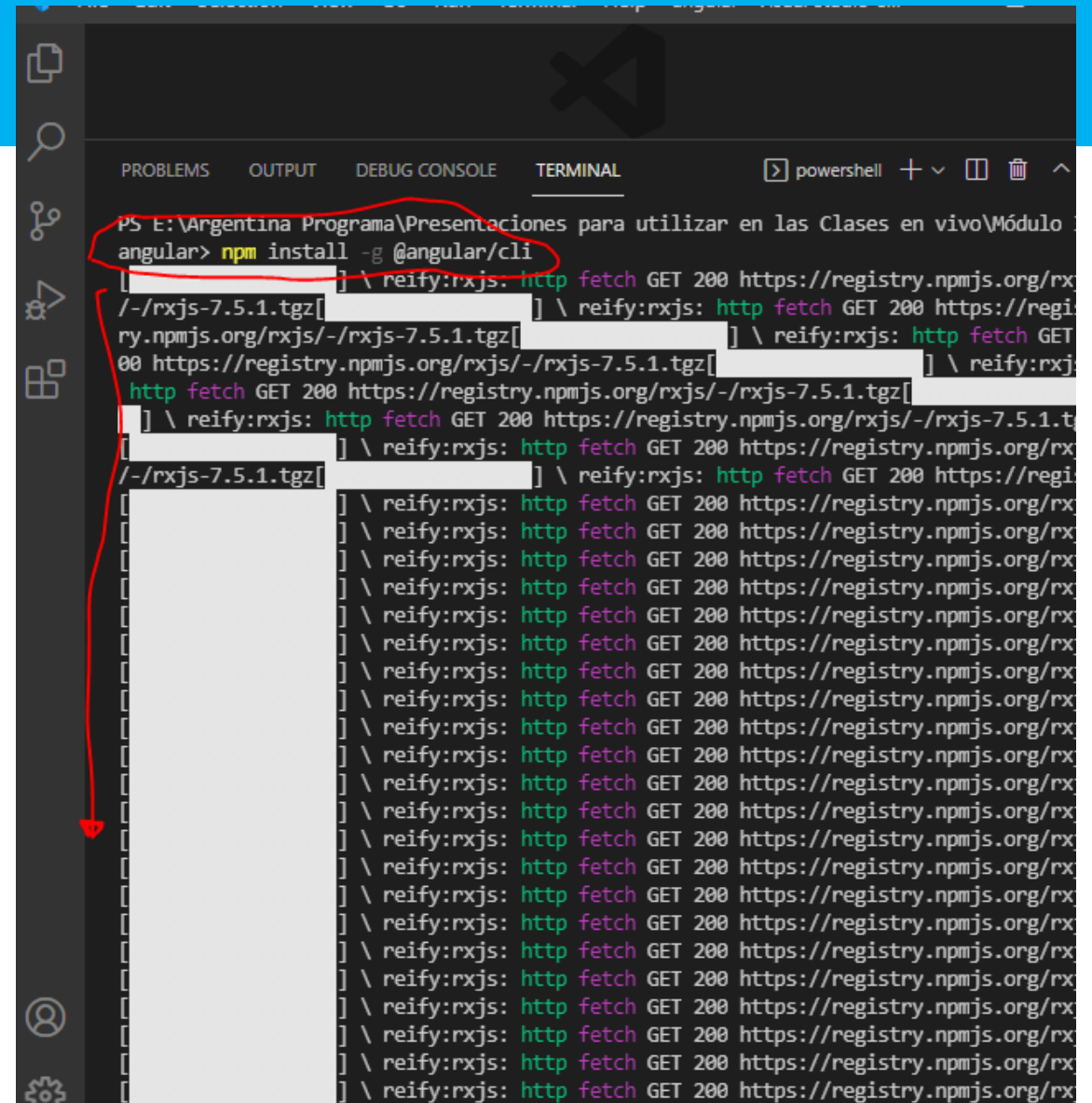
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Instalación De Angular

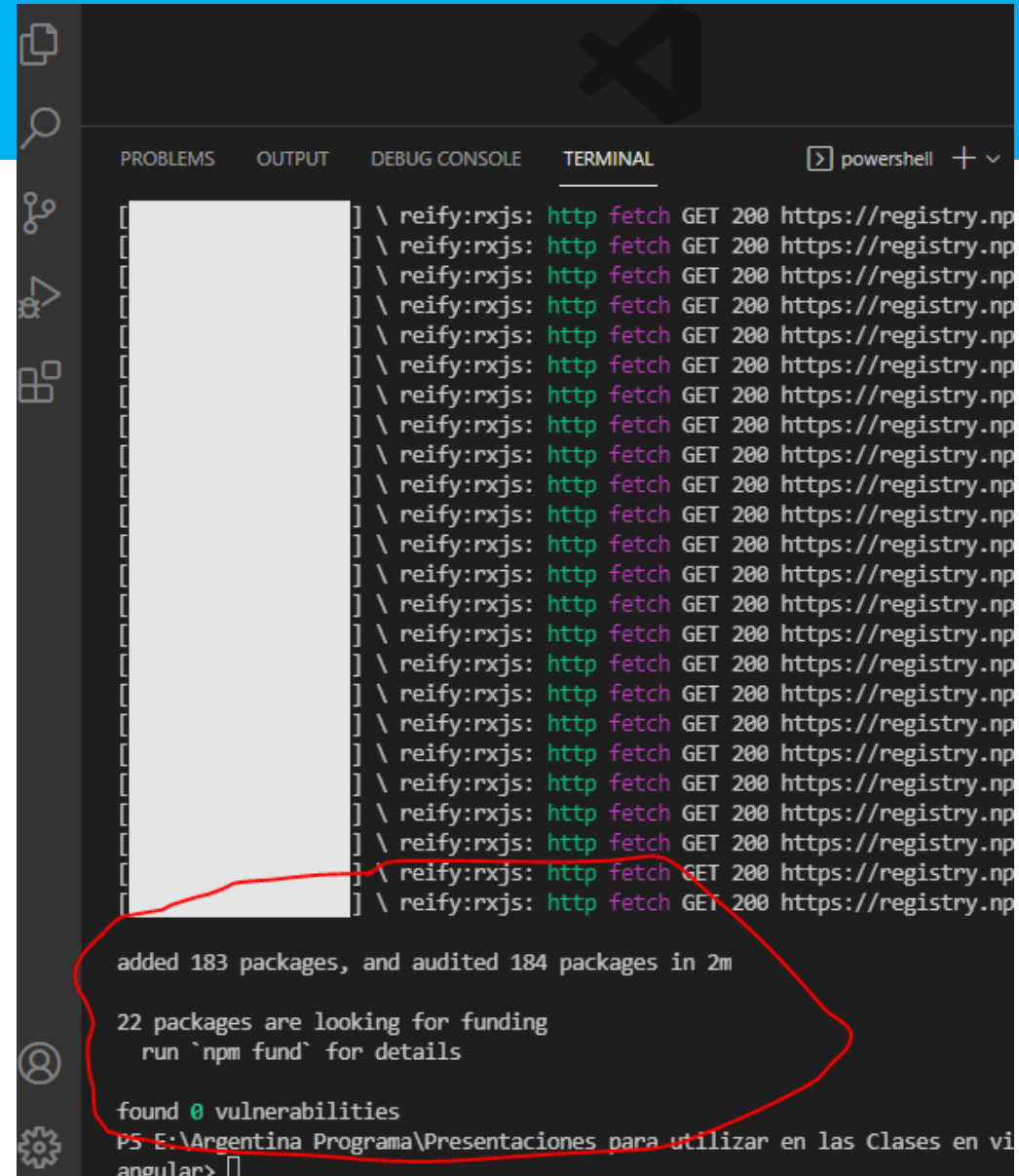
- Ejecutando el siguiente comando sobre la terminal del VSCode para iniciar el proceso de instalación de nuestro Angular para que sea reconocido por todos nuestros proyectos

**npm install -g @angular/cli**

```
angular> npm install -g @angular/cli
```



```
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 4 - Front End - Desarrollo Web Dinámico> angular> npm install -g @angular/cli
[...]
```



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Primera Aplicación Angular (Instalación en VSCode)

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Creando 1° App

- Ejecutando el siguiente comando sobre la terminal del VSCode nos permitirá generar nuestra primera App en Angular y se nos pedirá una serie de configuraciones para nuestro App.

**ng new {nombre de nuestra app}**


```
PS E:\Argentina Programa\Presentaciones pa  
angular> ng new miprimeraapp
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Creando 1° App

- En nuestra primera App no vamos a requerir Routing por consiguiente respondemos No (N)

```
PS E:\Argentina Programa\Presentaciones para utilizar en las C  
angular> ng new miprimeraapp
```



```
E:\Argentina Programa\Presentaciones para utilizar en las C  
angular> ng new miprimeraapp  
? Would you like to add Angular routing? (y/N) N
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Creando 1° App

- La siguiente pregunta que se nos realiza es sobre el tipo de formato que utilizaremos para manejar nuestro estilo de nuestra App, por defecto esta seleccionado CSS pero uno puede elegir la que desee moviendo las teclas direccionadoras y luego presionando intro.

```
angular> ng new miprimeraapp
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS [ https://sass-lang.com/documentation/syntax#scss ]
  Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less [ http://lesscss.org ]
```



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Creando 1° App

- Finalmente se procesa toda la información solicitada y se inicia el proceso de generación de nuestro proyecto para nuestra primera App de Angular

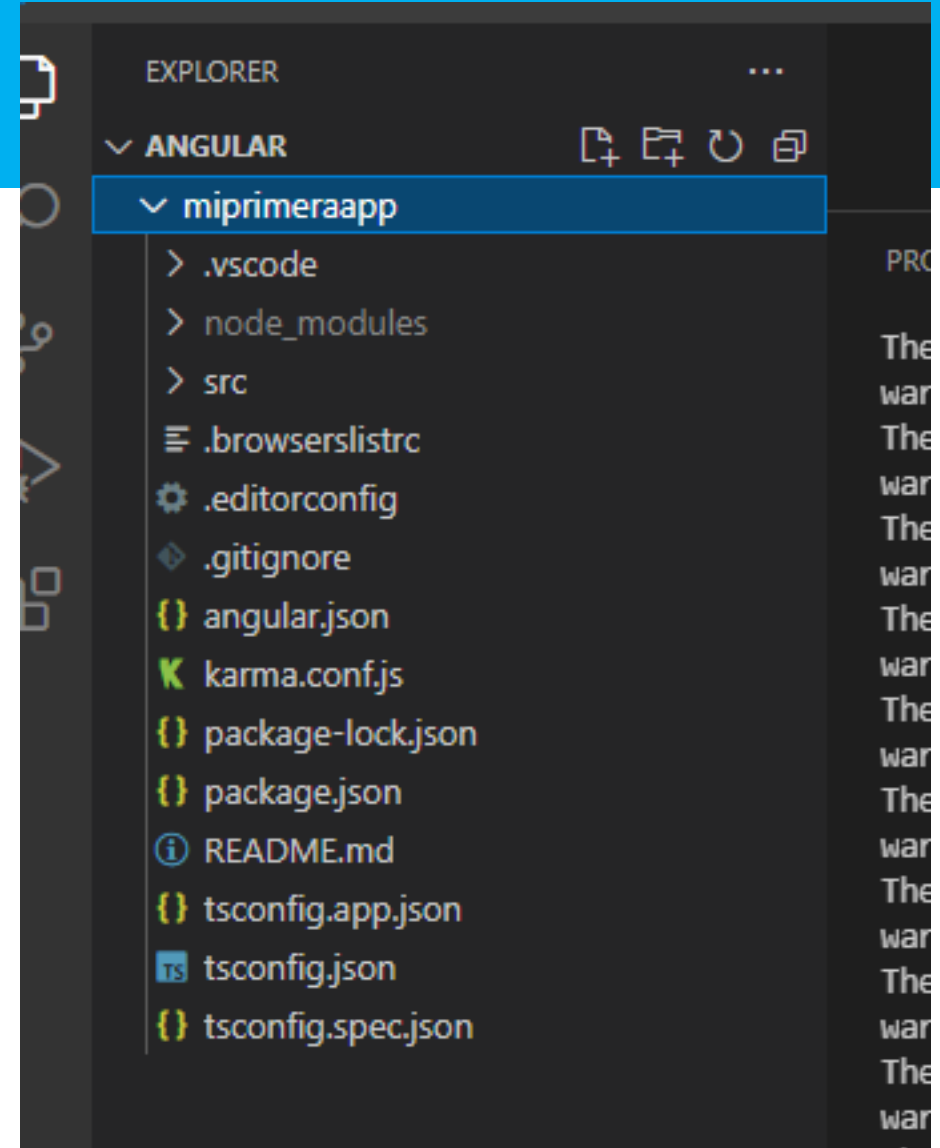
```
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/main.ts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/polyfills.ts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/styles.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/test.ts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tsconfig.app.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tsconfig.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tsconfig.spec.json.
The file will have its original line endings in your working directory
Successfully initialized git.
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular>
```

```
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular> ng new miprimeraapp
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE miprimeraapp/angular.json (3075 bytes)
CREATE miprimeraapp/package.json (1076 bytes)
CREATE miprimeraapp/README.md (1058 bytes)
CREATE miprimeraapp/tsconfig.json (863 bytes)
CREATE miprimeraapp/.editorconfig (274 bytes)
CREATE miprimeraapp/.gitignore (620 bytes)
CREATE miprimeraapp/.browserslistrc (600 bytes)
CREATE miprimeraapp/karma.conf.js (1429 bytes)
CREATE miprimeraapp/tsconfig.app.json (287 bytes)
CREATE miprimeraapp/tsconfig.spec.json (333 bytes)
CREATE miprimeraapp/.vscode/extensions.json (130 bytes)
CREATE miprimeraapp/.vscode/launch.json (474 bytes)
CREATE miprimeraapp/.vscode/tasks.json (938 bytes)
  ATE miprimeraapp/src/favicon.ico (948 bytes)
  ATE miprimeraapp/src/index.html (298 bytes)
  ATE miprimeraapp/src/main.ts (372 bytes)
  ATE miprimeraapp/src/polyfills.ts (2338 bytes)
  ATE miprimeraapp/src/styles.css (80 bytes)
  ATE miprimeraapp/src/test.ts (745 bytes)
  ATE miprimeraapp/src/assets/.gitkeep (0 bytes)
  ATE miprimeraapp/src/environments/environment.prod.ts (51 bytes)
  ATE miprimeraapp/src/environments/environment.ts (658 bytes)
  ATE miprimeraapp/src/app/app.module.ts (314 bytes)
  ATE miprimeraapp/src/app/app.component.html (23332 bytes)
  ATE miprimeraapp/src/app/app.component.spec.ts (974 bytes)
  ATE miprimeraapp/src/app/app.component.ts (216 bytes)
  ATE miprimeraapp/src/app/app.component.css (0 bytes)
Installing packages (npm)...
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Creando 1° App

- Lo que se genera una vez que finaliza el proceso de generación de nuestro proyecto de nuestra primera App.
- Para visualizar dicha App de Angular en nuestro navegador se debe de ejecutar la siguiente línea de comando para poder levantar o inicializar nuestro servidor Angular.



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Iniciar Serve Angular

- Para iniciar el servidor se deberá de ejecutar la siguiente línea de comando estando dentro de la carpeta raíz de nuestro proyecto

**cd {carpeta raíz de app}**

**ng serve --open /**

**ng serve -o**



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays the file structure of the 'ANGULAR' project, with the 'miprimeraapp' folder expanded. The file list includes: .vscode, node\_modules, src, .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, and package-lock.json. The Terminal panel at the bottom shows a PowerShell session with the following commands and output:

```
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular> cd miprimeraapp
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular\miprimeraapp> ng serve --open
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Iniciar Serve



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular> cd miprimeraapp  
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular\miprimeraapp> ng serve --open  
✓ Browser application bundle generation complete.
```

Initial Chunk Files	Names	Raw Size
vendor.js	vendor	1.73 MB
polyfills.js	polyfills	339.25 kB
styles.css, styles.js	styles	212.51 kB
main.js	main	51.45 kB
runtime.js	runtime	6.86 kB
Initial Total		2.32 MB

```
Build at: 2022-01-09T02:41:54.215Z - Hash: 752975c151d3fb03 - Time: 108191ms
```

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
```

```
✓ Compiled successfully.  
✓ Browser application bundle generation complete.
```

```
5 unchanged chunks
```

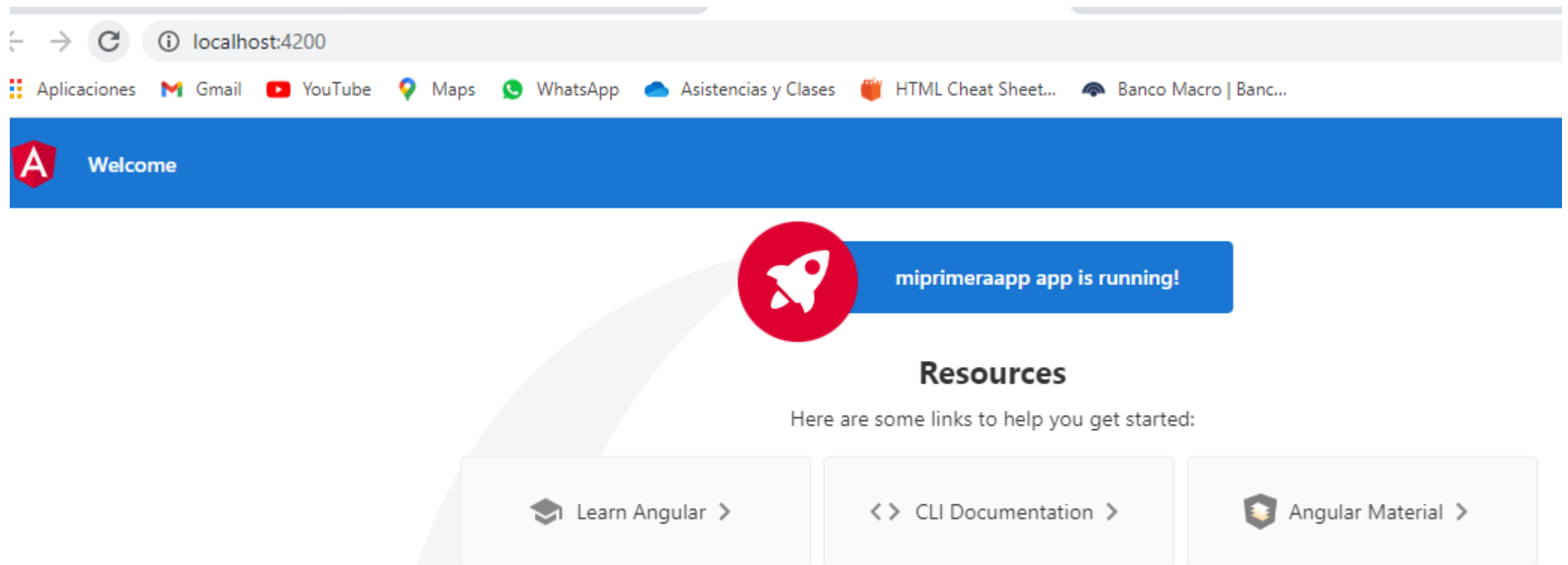
```
Build at: 2022-01-09T02:41:56.986Z - Hash: 752975c151d3fb03 - Time: 1187ms
```

```
✓ Compiled successfully.
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Iniciar Serve Angular

- La inicialización del servidor nos abre un navegador con la pag. de ejemplo de nuestra primera App



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Iniciar Serve Angular

Si por algún motivo no se abre nuestro servidor automáticamente, podemos abrir de la siguiente manera:

- Abrimos nuestro navegador favorito
- Ingresamos en la url la siguiente dirección como se nos indica en el sitio oficial de Angular

**`http://localhost:4200/`**

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Detener Serve Angular

Si por algún motivo deseamos detener el **SERVE** deberemos de hacer lo siguiente:

- Dentro de nuestra terminal, se observa nuestro servidor en ejecución, a continuación en donde se observa el cursor parpadeando presionamos **CTRL+C**, generando que se nos muestre nuevamente el pront con la ruta de nuestro directorio raíz y de esa manera se detiene nuestro **SERVE**.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular> cd miprimeraapp
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular\miprimeraapp> ng serve --open
✓ Browser application bundle generation complete.

Initial Chunk Files | Names          | Raw Size
vendor.js           | vendor         | 1.73 MB
polyfills.js        | polyfills      | 339.25 kB
styles.css, styles.js | styles        | 212.51 kB
main.js             | main           | 51.45 kB
runtime.js          | runtime        | 6.86 kB

| Initial Total | 2.32 MB

Build at: 2022-01-09T02:41:54.215Z - Hash: 752975c151d3fb03 - Time: 108191ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
✓ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2022-01-09T02:41:56.986Z - Hash: 752975c151d3fb03 - Time: 1187ms

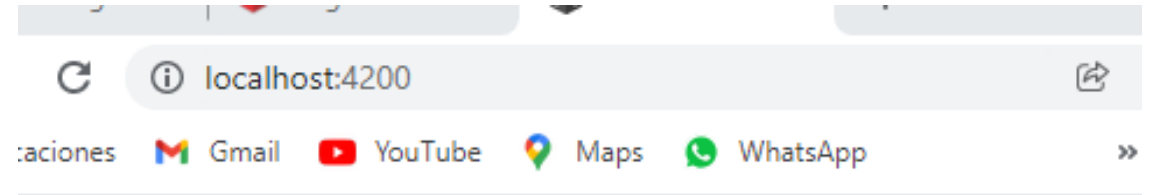
✓ Compiled successfully.

PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular\miprimeraapp>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Detener Serve Angular

Si volvemos a nuestro navegador y actualizamos la página nos tiene que dar un mensaje de que no se puede acceder al sitio solicitado.



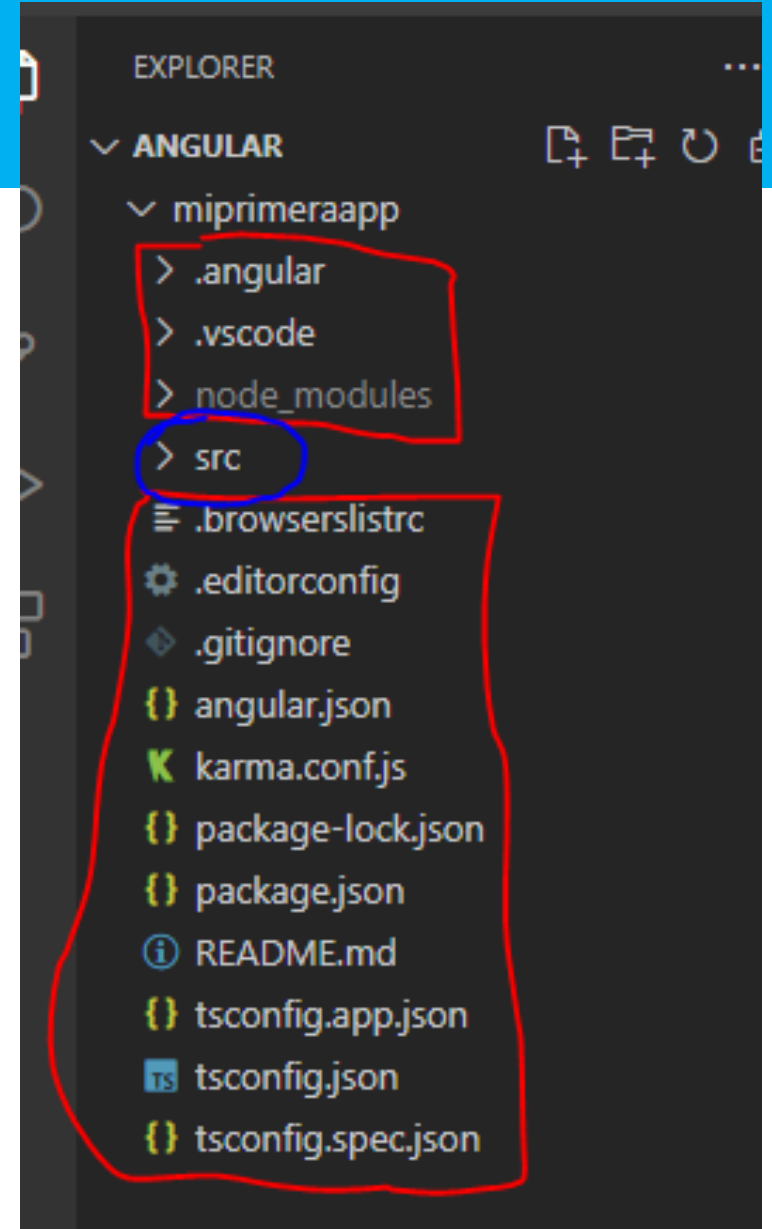
No se puede acceder a este sitio web



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Estructura App Angular

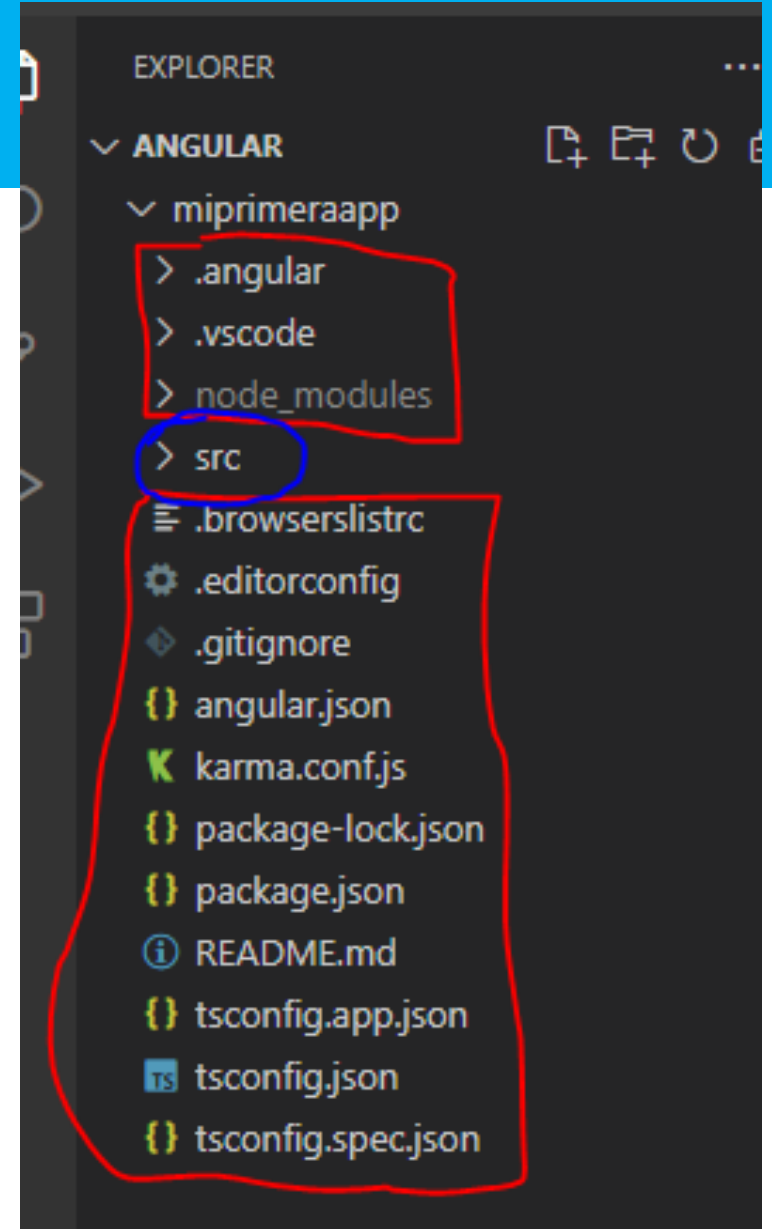
- Todas las carpetas salvo **SRC** y los archivos restantes serán con los que existen en la etapa de desarrollo y son archivos de configuración con los que trabaja Angular y solo existen en esa etapa de desarrollo.
- Una vez que la App ya se encuentra en su estado final o se busca publicarla en servidor se deberá de publicar todos que exista dentro de la carpeta **SRC**.
- Si desea modificar, agregar o eliminar información a la App se deberá de Trabajar en SRC.



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Estructura App Angular

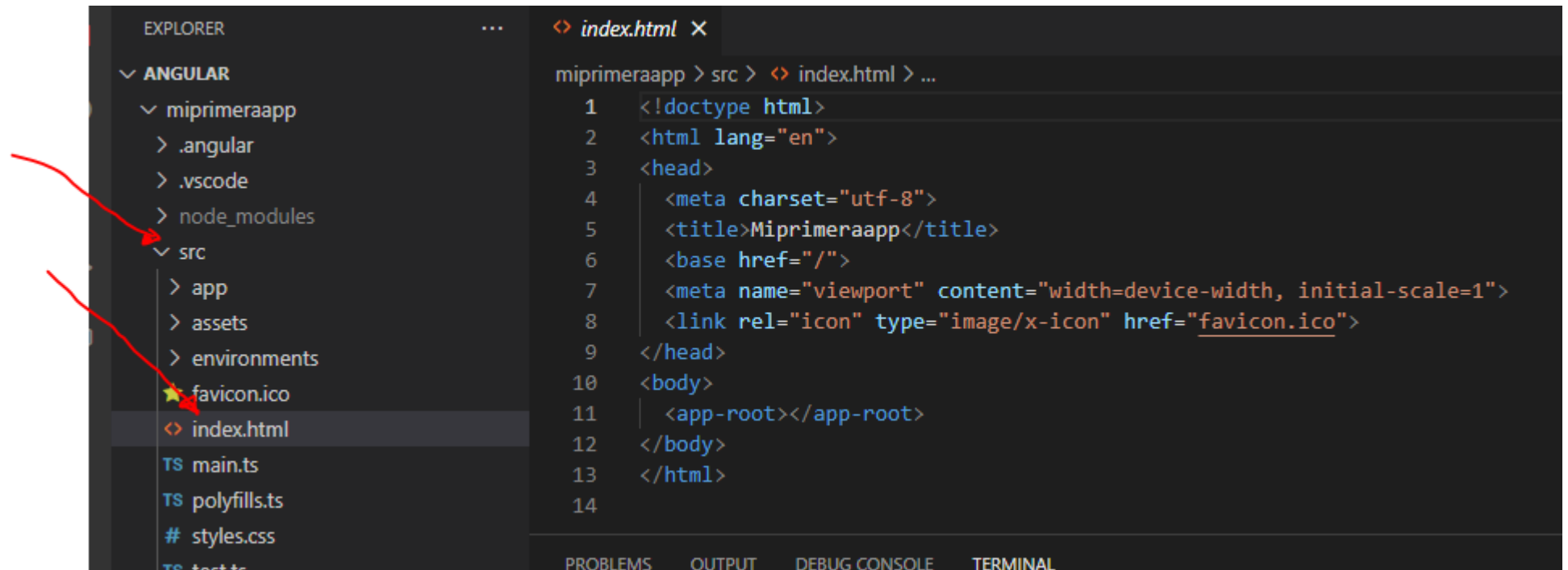
- Todas las carpetas salvo **SRC** y los archivos restantes serán con los que existen en la etapa de desarrollo y son archivos de configuración con los que trabaja Angular y solo existen en esa etapa de desarrollo.
- Una vez que la App ya se encuentra en su estado final o se busca publicarla en servidor se deberá de publicar todos que exista dentro de la carpeta **SRC**.
- Si desea modificar, agregar o eliminar información a la App se deberá de Trabajar en **SRC**.
- Dentro de **SRC** encontramos nuestro archivo index.html



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Modificando archivo index.html

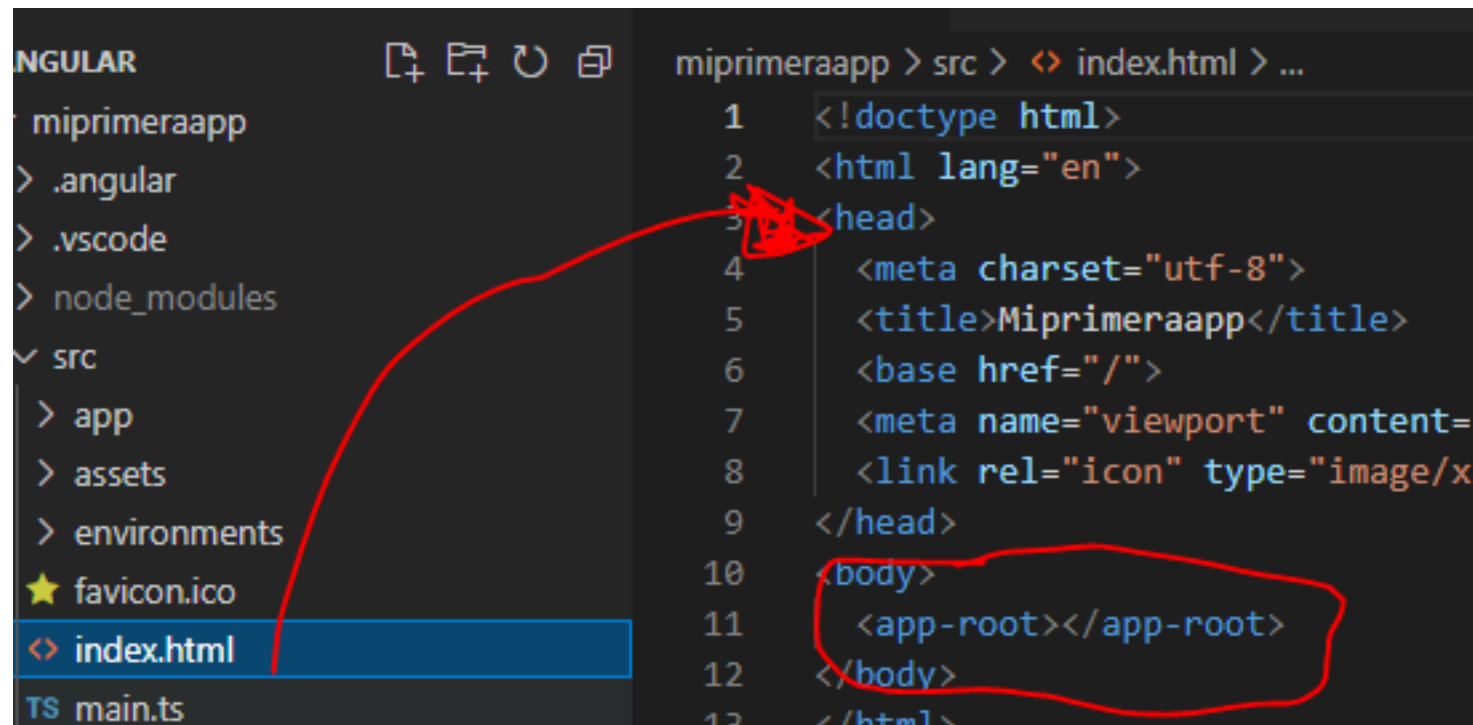
- Accedemos a la carpeta **SRC** y dentro de esta abrimos a index.html que es el archivo que se abre en nuestro navegador cuándo se levanta el **Serve**



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Modificando archivo index.html

- Se observa dentro de la carpeta **SRC** un archivo index.html, y al ingresar en este se observa que dentro de este hay una llamada o invocación a un componente que son la base de cualquier App de Angular y se llama **app-root**.



```
ANGULAR
miprimeraapp
> .angular
> .vscode
> node_modules
src
> app
> assets
> environments
★ favicon.ico
<> index.html
TS main.ts

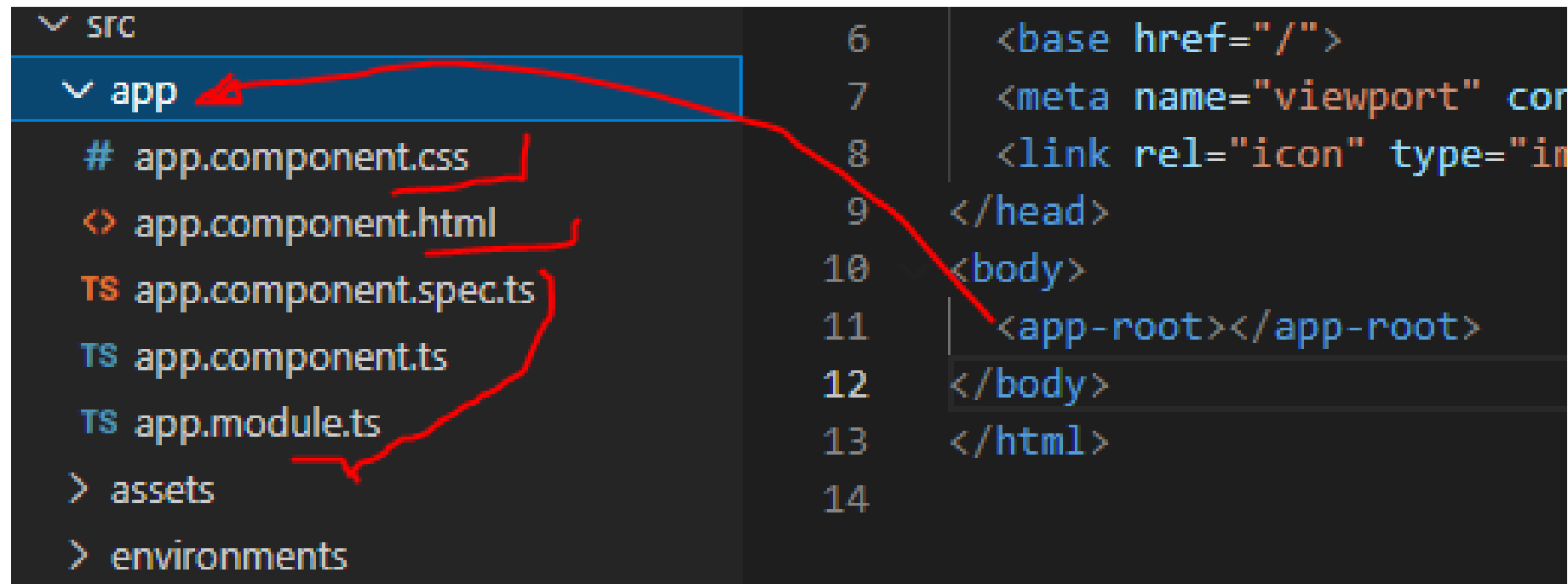
miprimeraapp > src > <> index.html > ...
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Miprimeraapp</title>
6 <base href="/">
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11 <app-root></app-root>
12 </body>
13 </html>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Modificando archivo index.html

- En el interior de index encontramos la referencia a un componente llamado app-root el cual esta asociado a la capeta app.

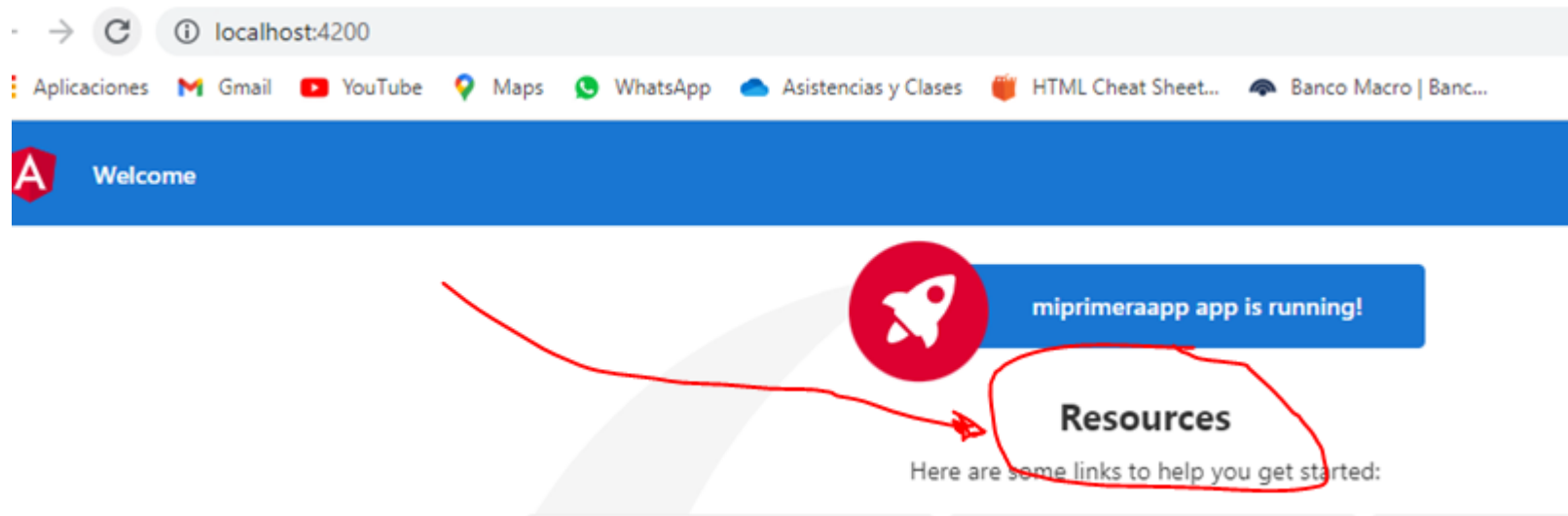
- En la carpeta app encontramos los siguientes Archivos:  
Css, Html y archivos de TypeScript



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Modificando archivo index.html

- En el interior del archivo app.component.html observamos que en su interior encontramos el código que se utiliza para mostrar lo que vemos en el sitio, código que ya viene predefinido para nuestro ejemplo.
- Tarea: Codifique el título **Resources** por su nombre

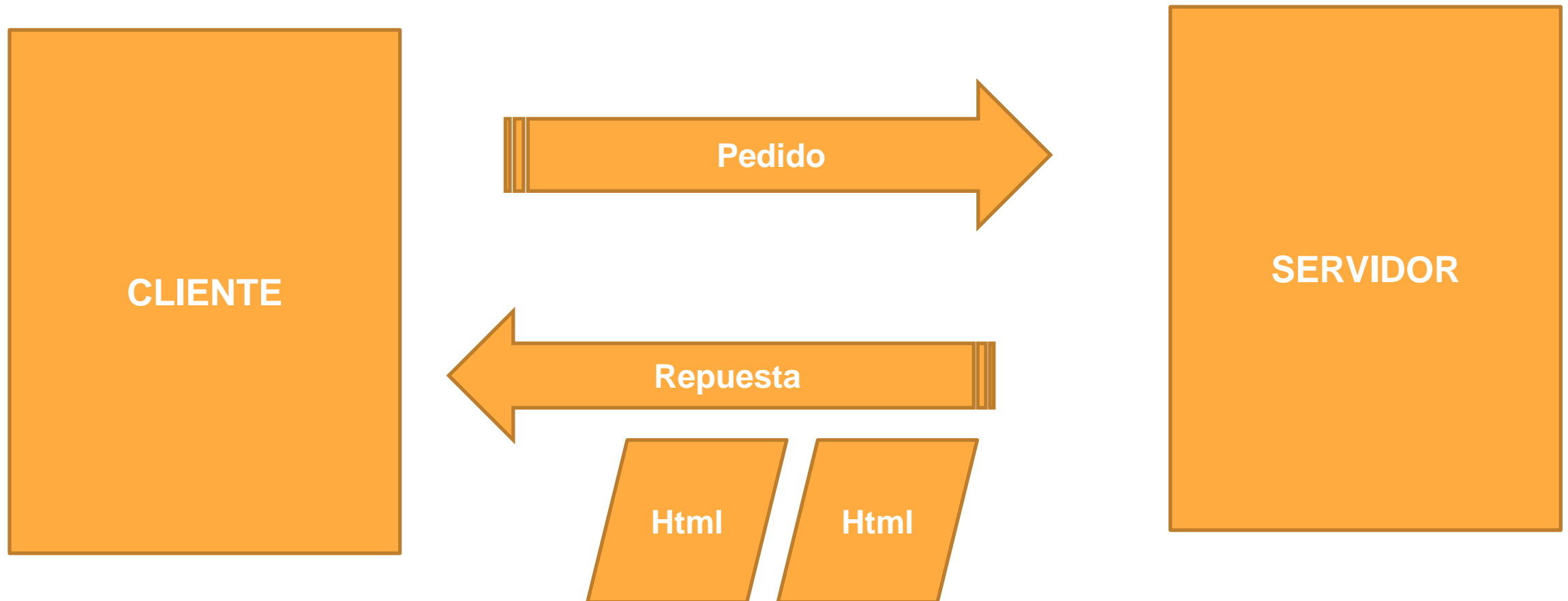


# Módulo 4 - Front End - Desarrollo Web Dinámico

## Estructura y Flujo de Ejecución en Aplicación Angular

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Procedimiento de comunicación entre cliente y servidor SPA





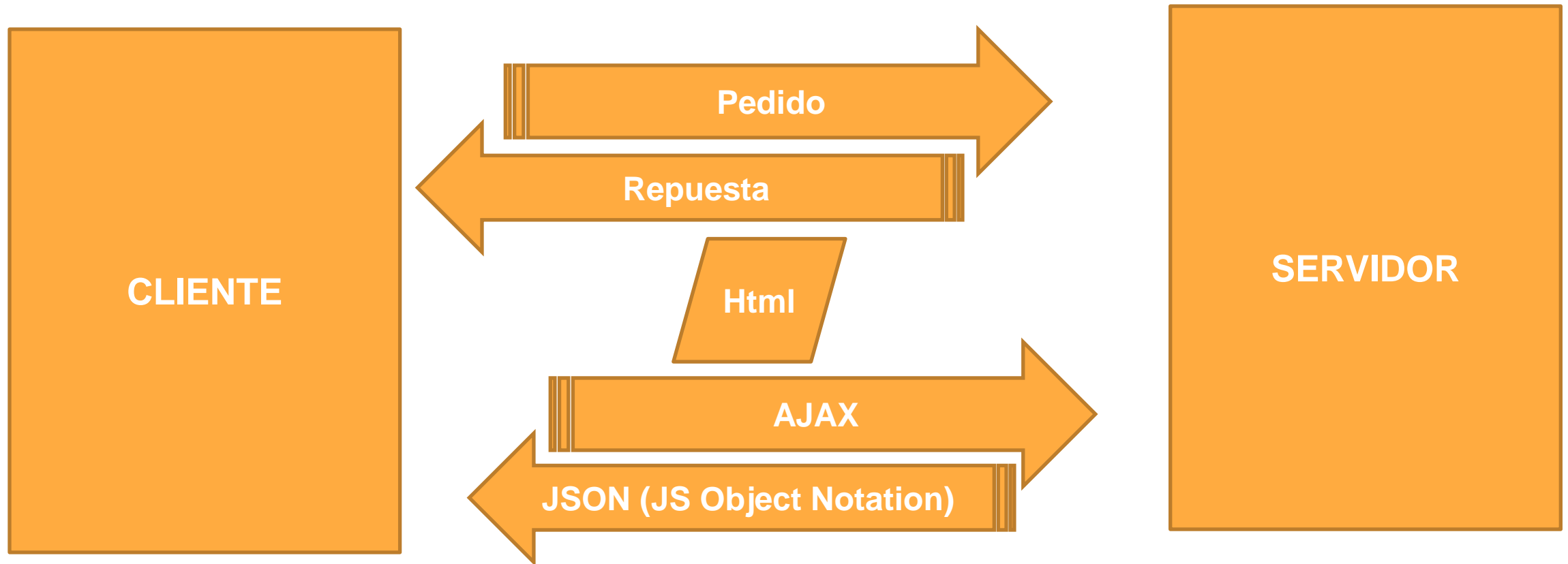
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Modificando archivo index.html

- En el proceso de solicitud y respuesta que se realiza entre el cliente y servidor se tienen que por cada solicitud que el cliente realiza existe un archivo html como respuesta a dicho pedido y esto se repite por cada pag html que se tenga.
- En una App se tiene que se realiza el pedido y la posterior carga de una pag html al inicio pero luego por cada pedido que realice el cliente al servidor de algún nuevo dato, se actualiza el sector o porción de la pag que se cargo inicialmente con lo cual en lugar de cargar toda la pag de nuevo solo se actualiza una parte de la misma.
- El seccionamiento y actualización de cada parte esta siendo trabajada mediante una tecnología llamada **Ajax** y las respuestas a los distintos ajax que genera el cliente nos llega desde el servidor en formato **JSON** o Notación de Objeto de JavaScript que son archivos de textos ligeros en donde se realizan los respectivos intercambios de datos entre servidor y cliente.

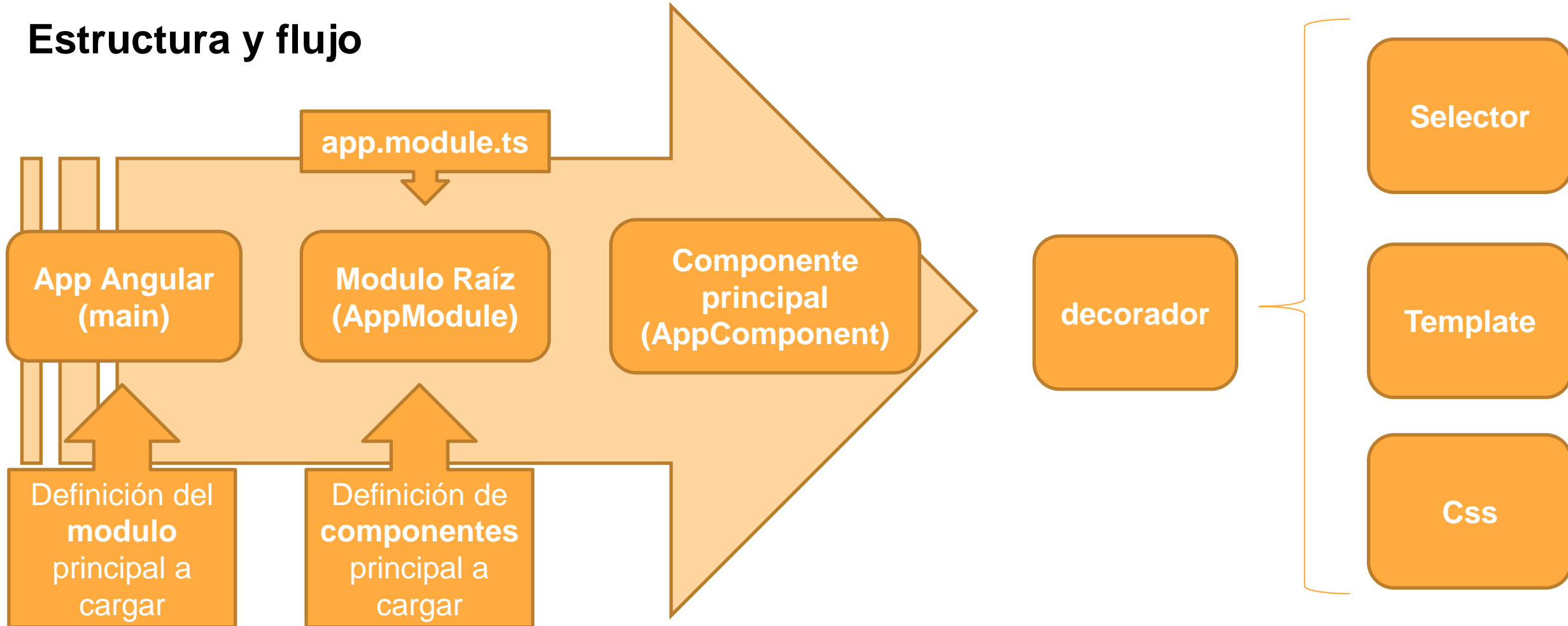
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Procedimiento de comunicación entre cliente y servidor



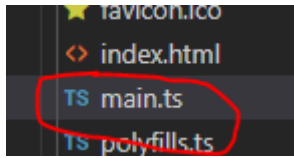
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Estructura y flujo



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Estructura y flujo



- El main es una función principal la cual se la utiliza para cargar un modulo principal por donde iniciara la aplicación.
- Este modulo principal o raíz esta definido en un archivo llamado **app.module.ts** el cual se encarga de la definición de los componentes a cargar.
- Por defecto el componente principal se llama **app.component.ts** el cual tiene en su interior la definición de una clase y un decorador.

```
import { AppModule } from './app/app.module';  
import { AppComponent } from './app.component';
```

```
4 import { AppComponent } from './app.component';  
5
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Estructura y flujo

- Un decorador no es más que un código javascript que define el comportamiento que tendrá la clase a la que acompaña como también así metadatos, tipificaciones, a su vez este lo componen un archivo css o style, un código ts y un selector.

```
import { Component } from '@angular/core'

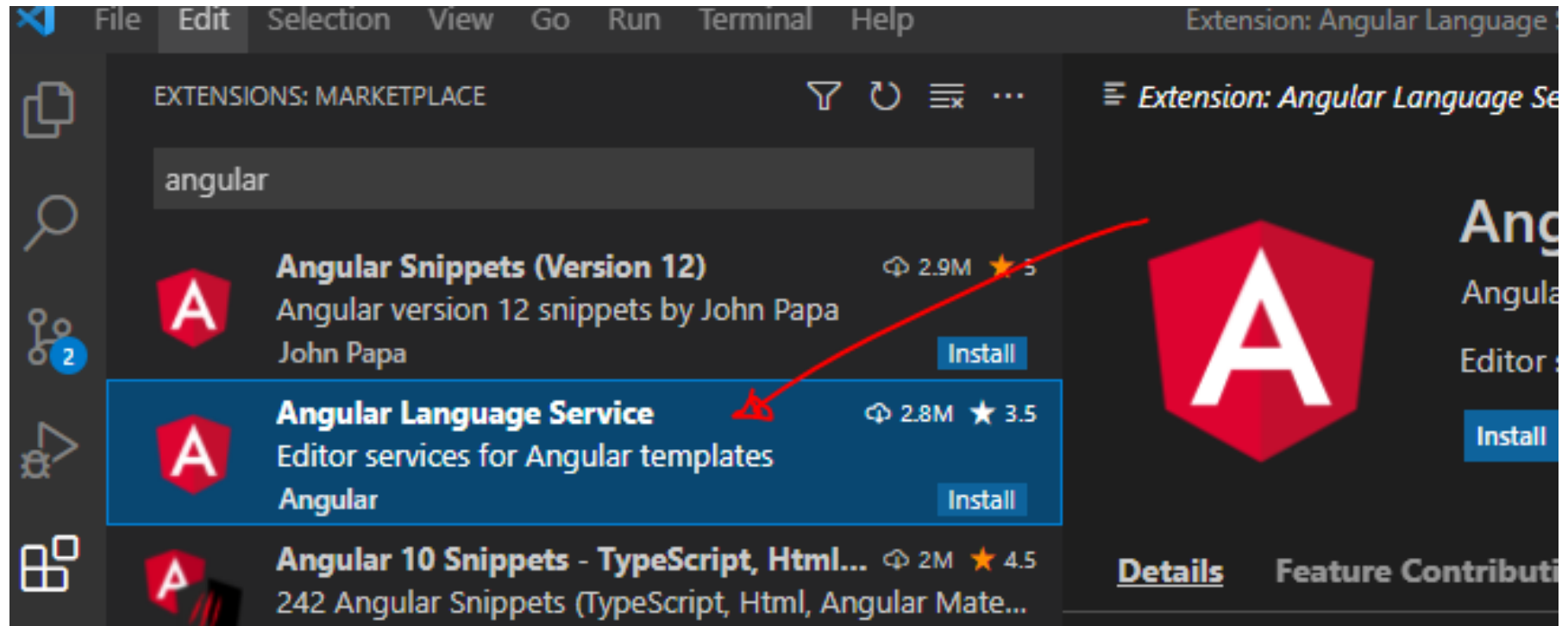
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Extensiones de Angular

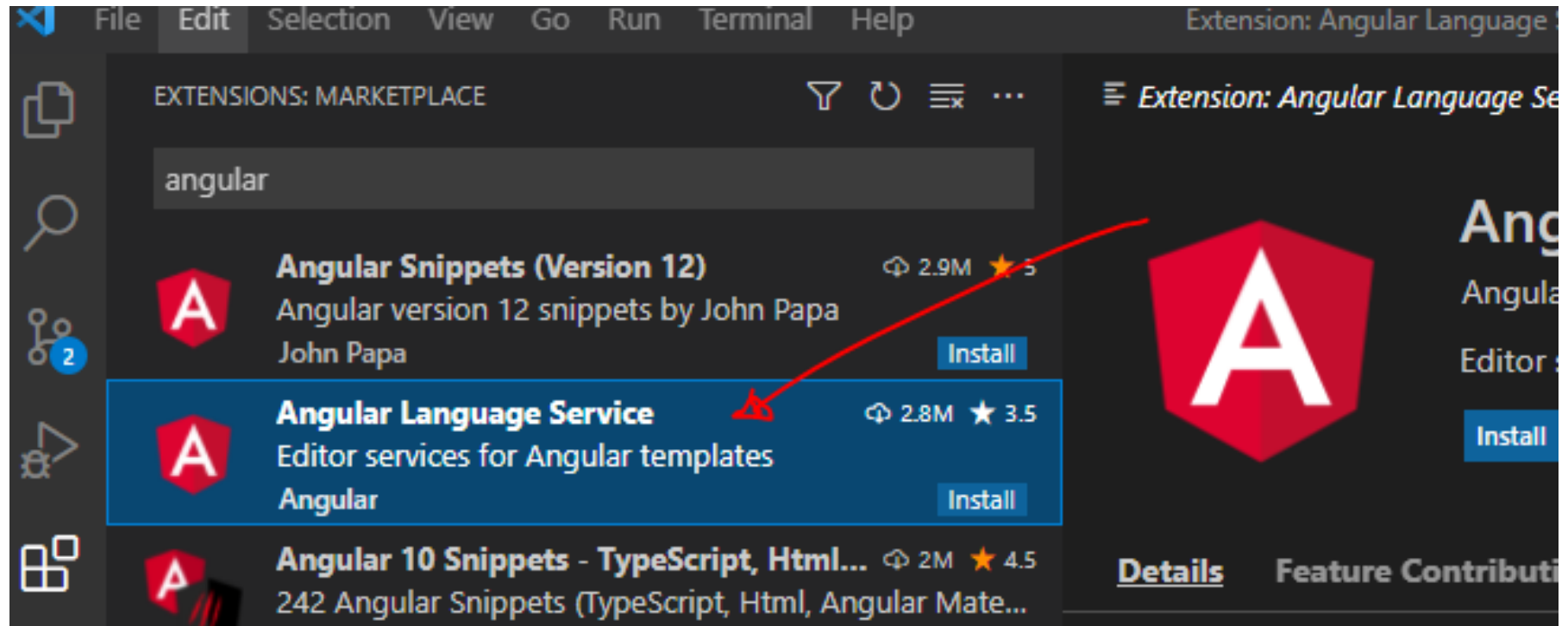
- Estas herramientas nos ofrece una mejor forma de ir aprendiendo Angular.



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Extensiones de Angular

- Estas herramientas nos ofrece una mejor forma de ir aprendiendo Angular.



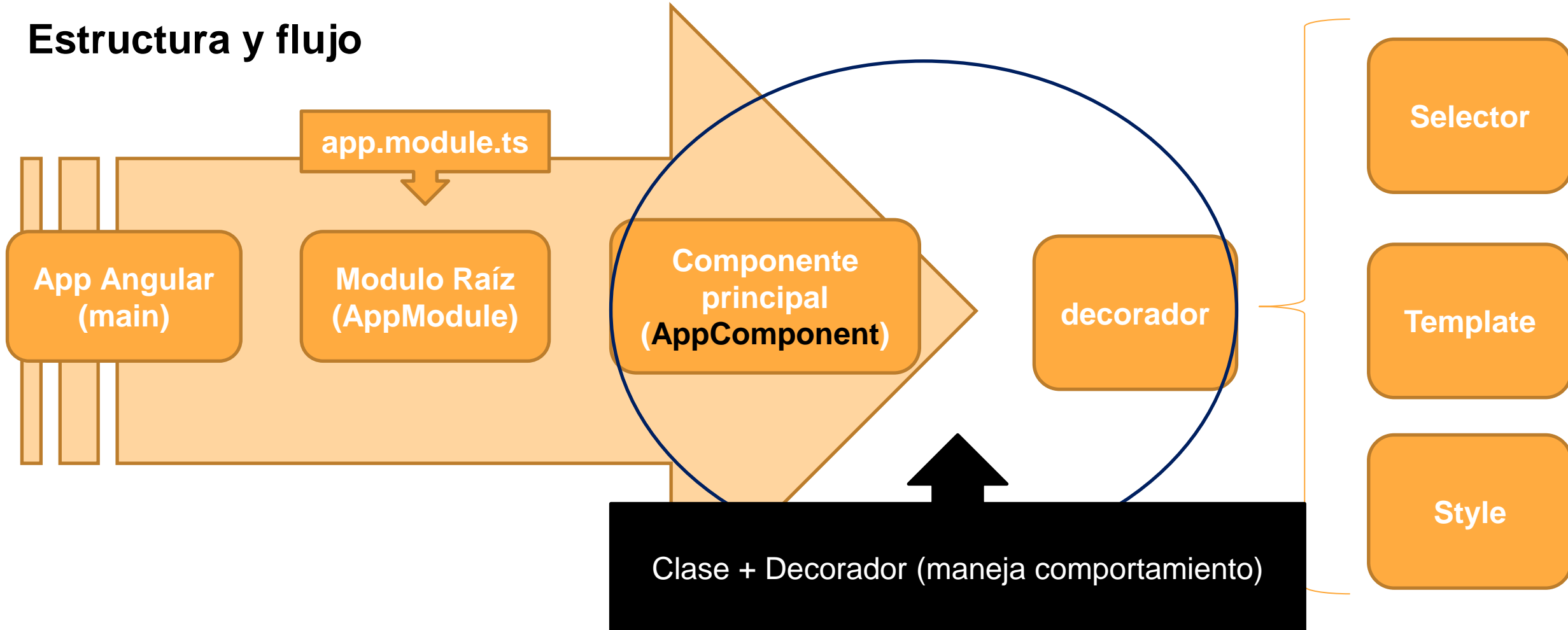
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Componentes en Aplicación Angular

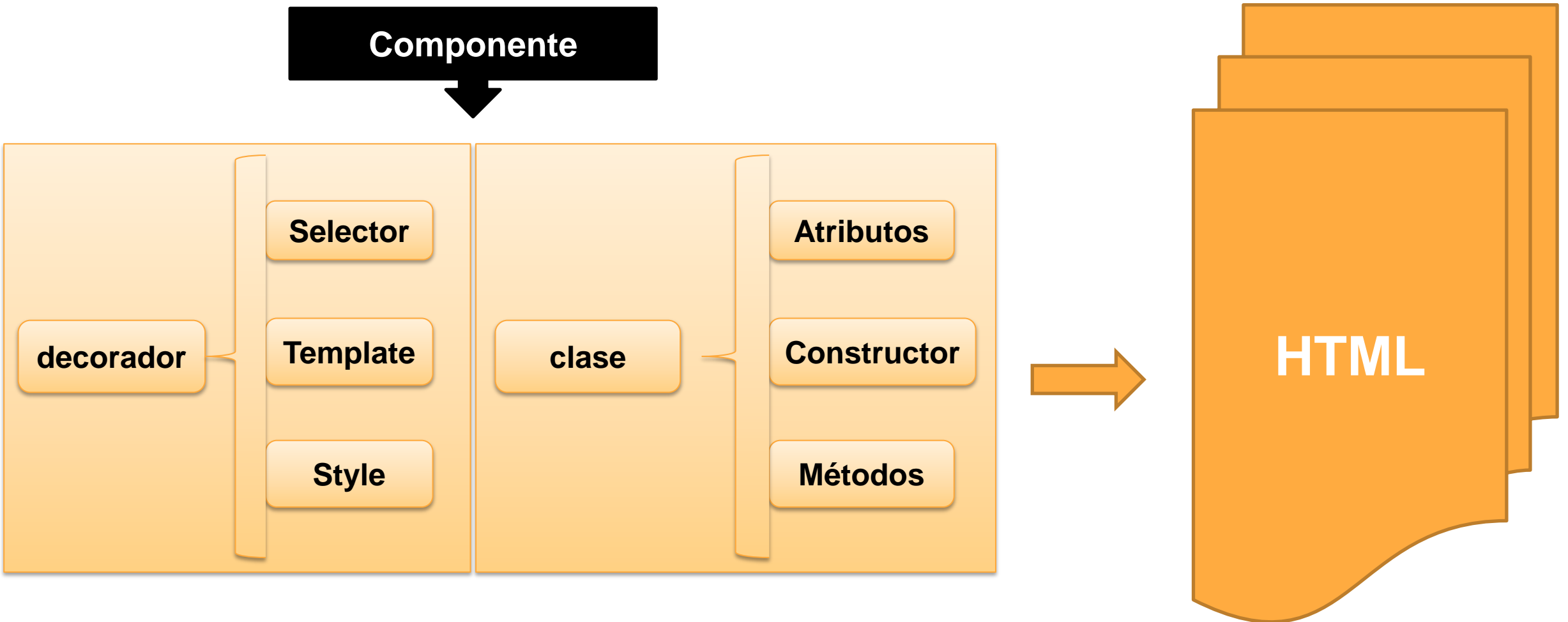


# Módulo 4 - Front End - Desarrollo Web Dinámico

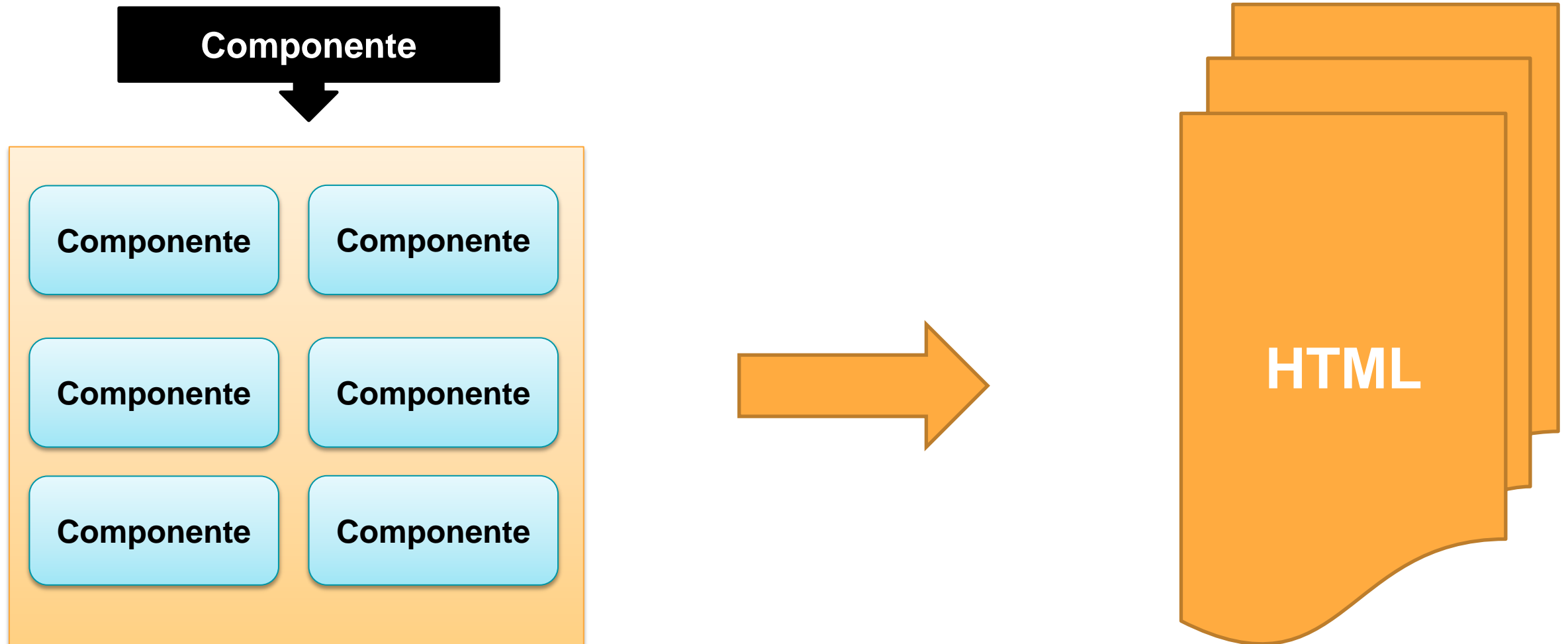
## Estructura y flujo



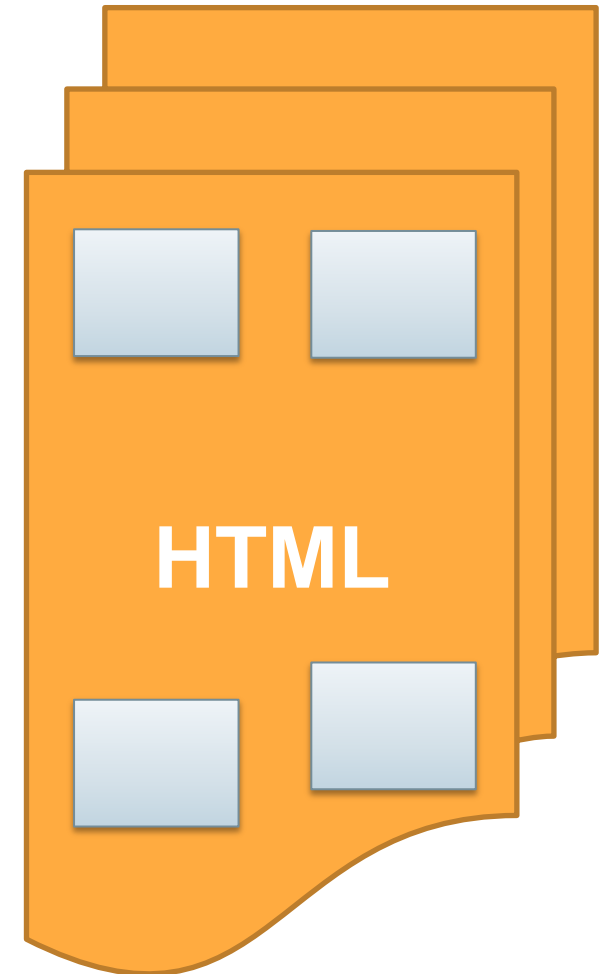
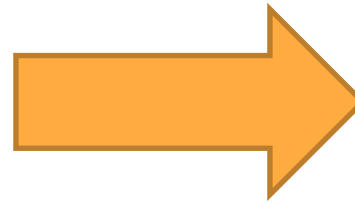
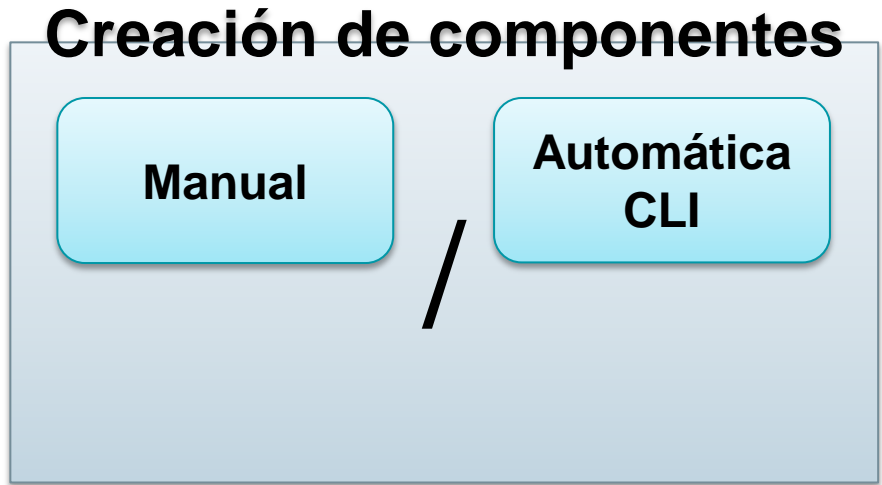
# Módulo 4 - Front End - Desarrollo Web Dinámico



# Módulo 4 - Front End - Desarrollo Web Dinámico



# Módulo 4 - Front End - Desarrollo Web Dinámico



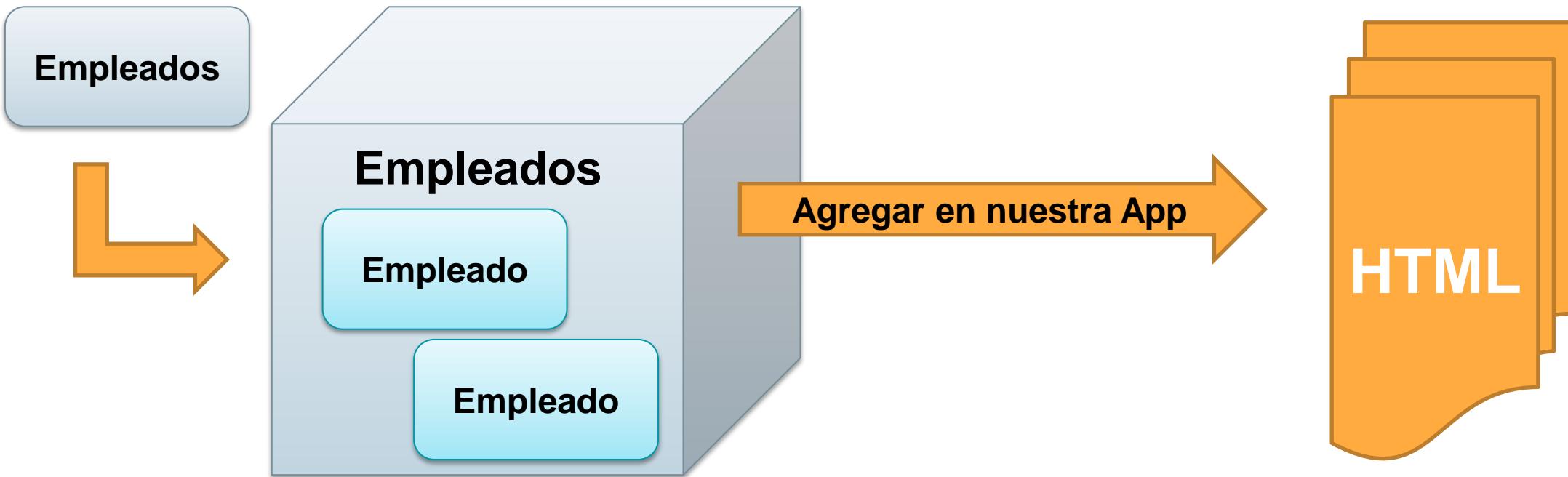
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Creando un Componente

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo. Creando un componente en Angular

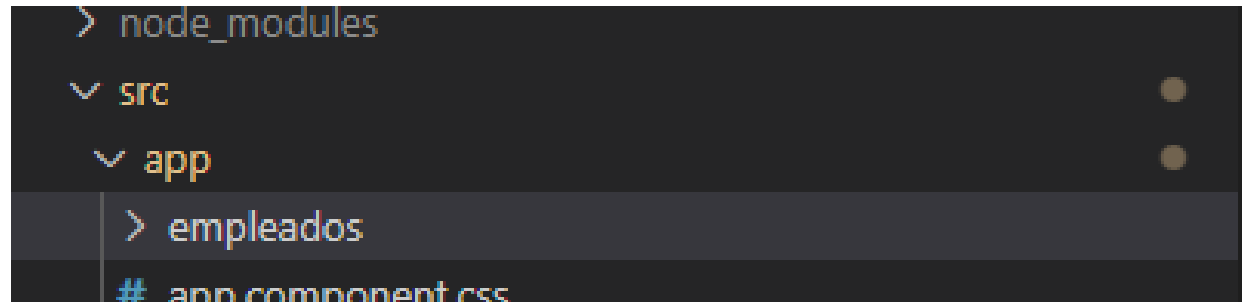
- El ejercicio va a consistir en que se creara un componente **Empleados** y dentro de este generaremos otro componente llamado **empleado**



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo. Creando un componente en Angular

- Comenzamos por generar un directorio **empleados** dentro de **src->app**.

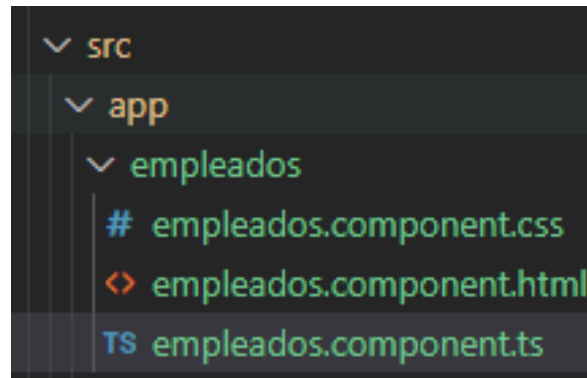


- A continuación generamos los siguientes archivos que corresponderán a nuestro componente raíz empleados, respetando la nomenclatura que plantea Angular tendríamos lo siguiente:

**empleados.component.css**

**empleados.component.html**

**empleados.component.ts**



# Módulo 4 - Front End - Desarrollo Web Dinámico

Dentro de **empleados.component.ts** redactamos el siguiente código:

//Segundo declaramos nuestro decorador que acompaña  
//a la clase

```
import { Component } from "@angular/core";
```

```
@Component({  
  //Tercero agregamos referencias a nuestra app  
  //empleados html  
  selector:'app-empleados',  
  templateUrl:'./empleados.component.html',  
  styleUrls:['./empleados.component.css']  
})
```

```
//Primero declaramos nuestra clase raiz empleados  
//export indicamos alcance global  
export class EmpleadosComponent {  
}
```

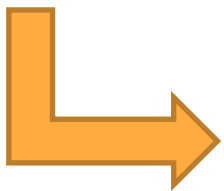
```
miprimeraapp > src > app > empleados > TS empleados.component.ts > EmpleadosComponent  
1  //Segundo declaramos nuestro decorador que acompaña  
2  //a la clase  
3  import { Component } from "@angular/core";  
4  
5  @Component({  
6    //Tercero agregamos referencias a nuestra app  
7    //empleados html  
8    selector:'app-empleados',  
9    templateUrl:'./empleados.component.html',  
10   styleUrls:['./empleados.component.css']  
11  })  
12  
13  //Primero declaramos nuestra clase raiz empleados  
14  //export indicamos alcance global  
15  export class EmpleadosComponent{  
16  
17  }
```



# Módulo 4 - Front End - Desarrollo Web Dinámico

Dentro de **empleados.component.html** redactamos el siguiente código:

```
<!--Agregamos el código html que se desea  
mostrar en el component -->  
<h1>Componente Raiz</h1>  
<h2>Empleados de la empresa</h2>
```



```
empleados.component.html U  
miprimeraapp > src > app > empleados > empleados.component.html > h2  
Go to component  
1 <!--Agregamos el código html que se desea  
2 mostrar en el component -->  
3 <h1>Componente Raiz</h1>  
4 <h2>Empleados de la empresa</h2>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

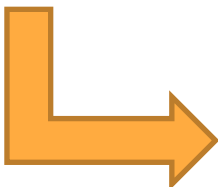
Dentro de empleados.component.css redactamos el siguiente código:

/\*Agregamos los respectivos estilos para las

etiquetas deseadas\*/

```
h1 {  
  color: blue;  
}
```

```
h2 {  
  color: red;  
}
```



```
empleados.component.html U # empleados.component.css U X  
primeraapp > src > app > empleados > # empleados.component.css > h1  
1  /*Agregamos los respectivos estilos para las  
2  etiquetas deseadas*/  
3  h1 {  
4    color: blue;  
5  }  
6  
7  h2 {  
8    color: red;  
9  }
```

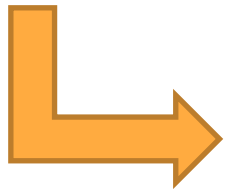
# Módulo 4 - Front End - Desarrollo Web Dinámico

**Registrar una App**

# Módulo 4 - Front End - Desarrollo Web Dinámico

Para poder utilizar un nuevo component en Angular se los debe de registrar y al mismo se lo hace dentro del archivo **app.module.ts** donde deberemos de agregar el siguiente código a continuación de la declaración de nuestro componente principal:

```
@NgModule({  
  declarations: [  
    AppComponent, EmpleadosComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
})
```



```
import { AppComponent } from './app.component';  
import { EmpleadosComponent } from './empleados/empleados.component';
```

```
@NgModule({  
  declarations: [  
    AppComponent, EmpleadosComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
})
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

A continuación para probar si nuestro componente raíz de **empleados** funciona lo agregaremos dentro de nuestro archivo principal de nuestra app llamado **app.component.html** donde deberemos de poder ver sobre el lienzo la programación que se realizó sobre este nuevo componente:

<!--Agregamos referencia a nuestro nuevo componente-->  
<app-empleados></app-empleados>



```
miprimeraapp > src > app > <> app.component.html > ...  
Go to component  
  
<!-- Resources -->  
<span style="color: red;"> Hola. El titulu  
  
<br/>  
  
<span>Saludos desde {{Saludo}}</span>  
  
<h2>Flores Cesar</h2>  
  
<br>  
  
<!--Agregamos referencia a nuestro nuevo comp  
<app-empleados></app-empleados>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

**Generando componente  
empleado que depende de  
componente raiz empleados**

# Módulo 4 - Front End - Desarrollo Web Dinámico

Para automatizar nuestra generación de los componentes lo que se hará es automatizar dicha tarea mediante la línea de comando CLI.

- Abriremos una nueva terminal dentro de nuestro VSCode

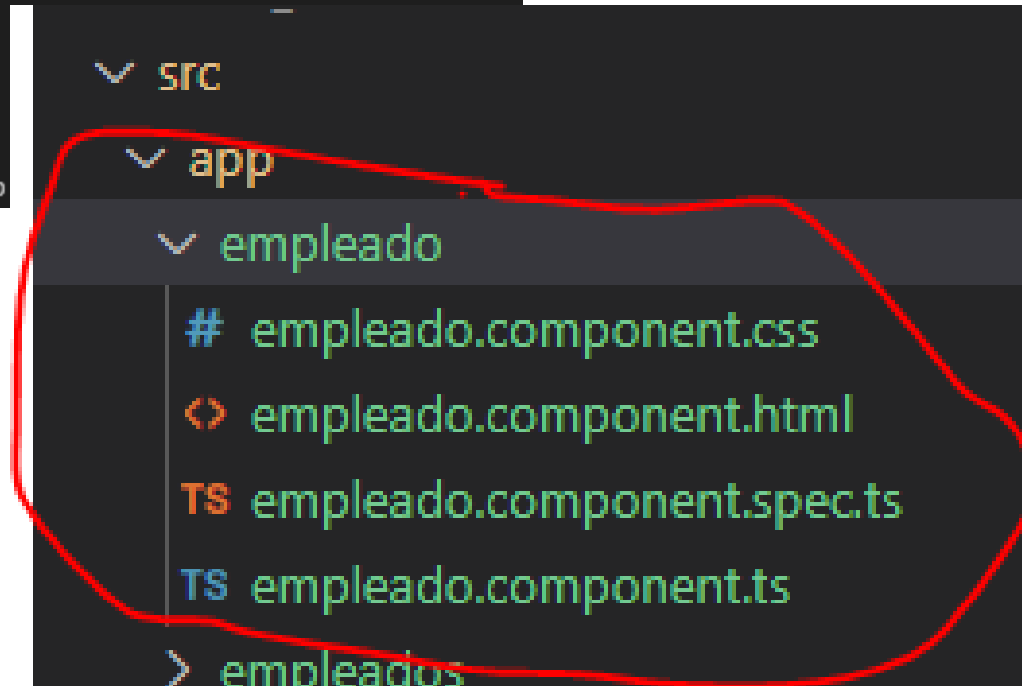


- luego deberemos pararnos dentro de nuestro directorio raíz mediante el comando **cd miprimeraapp**
- ejecutaremos el siguiente comando **ng generate component empleado** donde empleado es el nombre de nuestro nuevo componente, también podríamos abreviar dicho comando de la siguiente manera **ng g c empleado** ambas formas son validad.

# Módulo 4 - Front End - Desarrollo Web Dinámico

Al finalizar la ejecución de dicho comando podremos observar en nuestro árbol de proyecto un nuevo directorio llamado empleado con sus correspondientes archivos y que son todos los archivos que constituyen un componente:

```
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular> cd miprimeraapp
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo 3\angular\miprimeraapp> ng generate
component empleado
CREATE src/app/empleado/empleado.component.html (23 bytes)
CREATE src/app/empleado/empleado.component.spec.ts (640 bytes)
CREATE src/app/empleado/empleado.component.ts (283 bytes)
CREATE src/app/empleado/empleado.component.css (0 bytes)
UPDATE src/app/app.module.ts (491 bytes)
PS E:\Argentina Programa\Presentaciones para utilizar en las Clases en vivo\Módulo
```





# Módulo 4 - Front End - Desarrollo Web Dinámico

Adaptamos nuestro archivo html del nuevo componente **empleado.component.html** y posteriormente lo deberé de incluir dentro de nuestro archivo html de nuestro componente raíz de Empleados llamado **empleados.component.html** ya que dijimos que empleado depende del componente empleados:

empleado.component.html U X

miprimeraapp > src > app > empleado > empleado.component.html > ..

Go to component

```
1 <p>Aquí se genero el componente empleado
2 que depende del componente raíz Empleados</p>
```



empleados.component.html U X

miprimeraapp > src > app > empleados > empleados.component.html > ..

Go to component

```
1 <!--Agregamos el código html que se desea
2 mostrar en el component -->
3 <h1>Componente Raíz</h1>
4 <h2>Empleados de la empresa</h2>
5 <br>
6 <p>Se agrega un empleado nuevo</p>
7 <app-empleado></app-empleado>
8
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Interpolación



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Interpolación de Strings

- Para qué sirve
- Cómo se la utiliza.
- Ejemplo

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Interpolación – Para que se la utiliza

- Por medio de la interpolación se nos permite incorporar o agregar textos dinámicos, según la ayuda de ofrece el sitio oficial de Angular. <https://angular.io/guide/interpolation>.
- Dentro de una pag podemos tener dos tipos de textos, textos de tipo estáticos que no varían en el tiempo y textos que pueden responder ante algún tipo de acción o evento el cual puede cambiar en el tiempo, a estos se los conoce como textos dinámicos porque cambian durante la ejecución de nuestra App.
- Por ejemplo cuando nos logeamos y se nos muestra en algún sector de la pag el nombre del usuario logeado, ese texto es lo que se corresponde con un texto dinámico.
- La interpolación permite realiza operaciones de comparación lógicos, llamada a funciones entre otras cosas.


# Módulo 4 - Front End - Desarrollo Web Dinámico

## Interpolación – Creación atributos

- Agregaremos dentro de nuestra **clase empleado** que se encuentra dentro del archivo **empleado.component.ts** de nuestro **componente empleado** los datos referentes a *apellido, nombre, edad y profesión con sus respectivos métodos de lectura y escritura.*

```
export class EmpleadoComponent implements OnInit {  
  //Agregamos atributos a nuestra clase  
  //Agregamos atributos a nuestra clase  
  private vcapellido:string="Flores";  
  private vcnombre:string="Cesar";  
  private vcedad:number=40;  
  private vcprofesion:string="Programador";
```



```
TS empleado.component.ts U ●  
miprimeraapp > src > app > empleado > TS empleado.component.ts >   
4     selector: 'app-empleado',  
5     templateUrl: './empleado.component.html',  
6     styleUrls: ['./empleado.component.css']  
7   })  
8   export class EmpleadoComponent implements OnInit  
9     //Agregamos atributos a nuestra clase  
10    private vcapellido:string="Flores";  
11    private vcnombre:string="Cesar";  
12    private vcedad:number=40;  
13    private vcprofesion:string="Programador";  
14
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Interpolación – Creación métodos

```
//Metodos de lectura de atributos
public getApellido() { return this.vcapellido; }
public setApellido(dato:string) { this.vcapellido=dato; }
public getNombre() { return this.vcnombre; }
public setNombre(dato:string) { this.vcnombre=dato; }
public getEdad() { return this.vcedad; }
public setEdad(dato:number) { this.vcedad=dato; }
public getProfesion() { return this.vcprofesion; }
public setProfesion(dato:string) { this.vcprofesion=dato; }
```



```
1 private vcnombre:string="Cesar";
2 private vcedad:number=40;
3 private vcprofesion:string="Programador";
4
5 constructor() { }
6
7 //Metodos de lectura de atributos
8 public getApellido() { return this.vcapellido; }
9 public setApellido(dato:string) { this.vcapellido=dato; }
10 public getNombre() { return this.vcnombre; }
11 public setNombre(dato:string) { this.vcnombre=dato; }
12 public getEdad() { return this.vcedad; }
13 public setEdad(dato:number) { this.vcedad=dato; }
14 public getProfesion() { return this.vcprofesion; }
15 public setProfesion(dato:string) { this.vcprofesion=dato; }
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

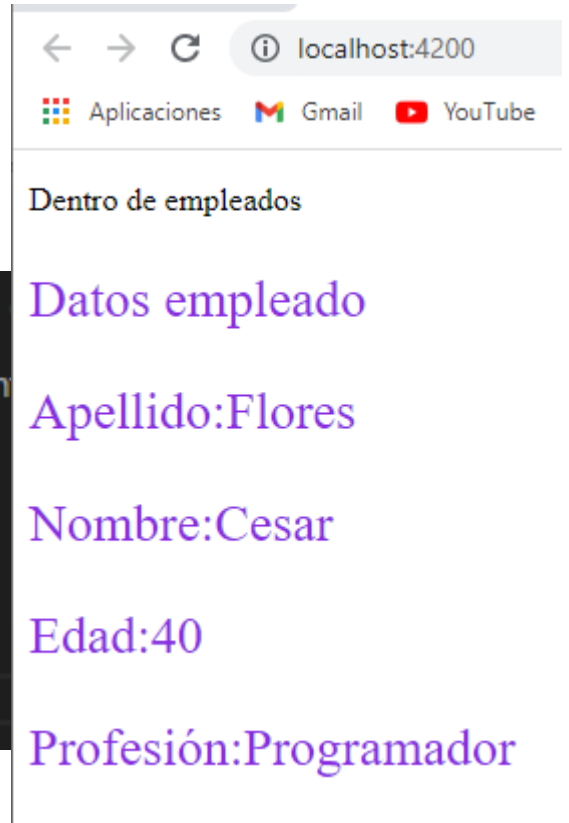
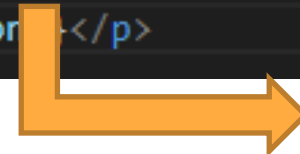
## Interpolación – Uso

- Agregaremos dentro de nuestro archivo template del componente **empleado.component.html** las referencia a nuestros atributos mediante interpolación:

```
<p>Datos empleado</p>
<p>Apellido:{{getApellido()}}</p>
<p>Nombre:{{getNombre()}}</p>
<p>Edad:{{getEdad()}}</p>
<p>Profesión:{{getProfesion()}}</p>
```



```
empleado.component.ts U empleado.component.html
primeraapp > src > app > empleado > empleado.component.html
Go to component
1 <p>Datos empleado</p>
2 <p>Apellido:{{getApellido}}</p>
3 <p>Nombre:{{getNombre}}</p>
4 <p>Edad:{{getEdad}}</p>
5 <p>Profesión:{{getProfesion}}</p>
```



# Módulo 4 - Front End - Desarrollo Web Dinámico

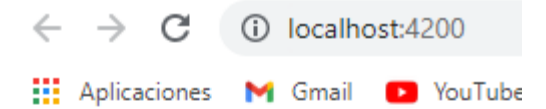
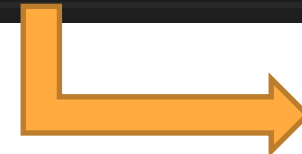
## Interpolación – Uso - Condicional

- Se indicara una condición que analizara la edad del empleado y mostrara un mensaje si es mayor de edad al lado de la edad del empleado, de esta manera se tiene otro ejemplo de interpolación:

```
<p>Edad:{{getEdad()<18 ? getEdad()+". Es menor":getEdad()+". Es mayor"}}</p>
```



```
<p>Datos empleado</p>
<p>Apellido:{{getApellido()}}</p>
<p>Nombre:{{getNombre()}}</p>
<p>Edad:{{getEdad()<18 ? getEdad()+". Es menor":getEdad()+". Es mayor"}}</p>
<p>Profesión:{{getProfesion()}}</p>
```



Dentro de empleados

Datos empleado

Apellido:Flores

Nombre:Cesar

Edad:40. Es mayor

Profesión:Programador



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Ingresar la edad desde un control text

- Se nos debe de pedir la edad del empleado y luego de presionar un botón se deberá de actualizar dinámicamente el atributo edad por su correspondiente mensaje de si es mayor de edad o no y mostrar la edad que se ingreso en el text.
- Antes que nada debemos de agregar dos controles nuevos dentro de nuestro html **empleado.component.html**, un control text de tipo número y un botón quien es el responsable de iniciar el evento de lectura y análisis del dato.

```
<!--Cuadro de texto de tipo numero para la edad-->
```

```
<p>Ingrese la edad del empleado
```

```
  <input type="number" name="" id=""  
min="1" max="100" #txtEdad>
```

```
  <button>Cambiar</button>
```

```
</p>
```

```
empleado.component.html U ●  
primeraapp > src > app > empleado > <> empleado.component.html > ...  
  Go to component  
1  <!--Cuadro de texto de tipo numero para la edad-->  
2  <p>Ingrese la edad del empleado  
3    <input type="number" name="" id="" min="1" max="100" #txtEdad>  
4    <button>Cambiar</button>  
5  </p> |
```

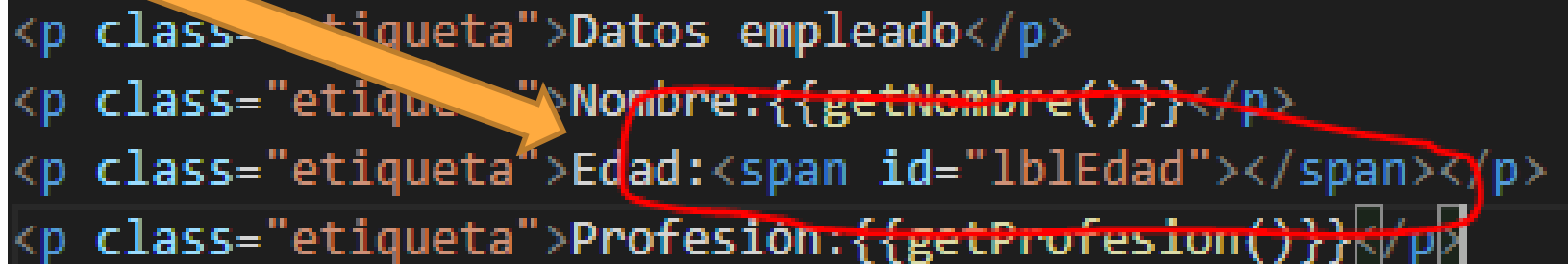
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Ingresar la edad desde un control text

- Agregado una etiqueta **span** con un **id='lblEdad'** que nos permitirá luego cambiar su contenido según lo que se ingrese en el control input llamado **#txtEdad**

```
<p class="etiqueta">Edad:<span id="lblEdad"></span></p>
```

```
<p>Datos empleado</p>
<p>Apellido:{{getApellido()}}</p>
<p>Nombre:{{getNombre()}}</p>
<p>Edad:{{getEdad()<18 ? getEdad()+". Es menor":getEdad()+". Es mayor"}}</p>
<p>Profesión:{{getProfesion()}}</p>
```



```
<p class="etiqueta">Datos empleado</p>
<p class="etiqueta">Nombre:{{getNombre()}}</p>
<p class="etiqueta">Edad:<span id="lblEdad"></span></p>
<p class="etiqueta">Profesión:{{getProfesion()}}</p>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Ingresar la edad desde un control text

- Dentro de archivo **empleado.component.ts** tenemos nuestra clase empleado modificamos el evento **getEdad** por el siguiente código:

```
public getEdad() { return this.vcedad <18 ? this.vcedad+". Es menor!!!":this.vcedad+". Es mayor!!!"; }
```

- Generamos una función que será la que procesara la edad que se ingresa desde el cuadro de texto y es la función que utiliza el botón para lanzar todo el proceso de carga y validación:

```
public AnalizarEdad(dato:string) {  
    //El dato que viene como parametro se lo paso a método setEdad para  
    //que cargue el atributo vcedad, se lo convierte a número  
    this.setEdad(parseInt(dato));  
    //Se muestra el resultado del método getEdad sobre el control span  
    (<HTMLOptionElement>document.getElementById("lblEdad")).innerText=this.getEdad();  
}
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Ingresar la edad desde un control text

- Realizaremos el agregado del evento click al botón con la llamada a la función **AnalizarEdad** que se encuentra dentro de la clase de empleado contenida en el archivo **empleado.component.ts**.


<!--Cuadro de texto de tipo numero para la edad-->

<p>Ingrese la edad del empleado

<input type="number" name="" id="" #txtEdad>

<button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>

</p>

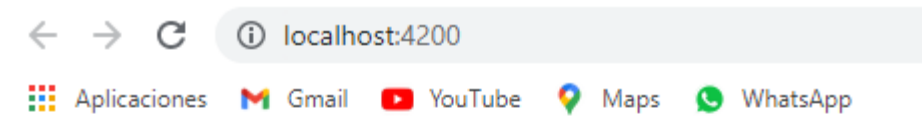


```
<input type="number" name="" id="" #txtEdad>
<button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Ingresar la edad desde un control text

- La página finalmente quedara de la siguiente manera



Dentro de empleados

Ingrese la edad del empleado  Cambiar

Datos empleado

Nombre:Cesar

Edad:

Profesión:Programador



Dentro de empleados

Ingrese la edad del empleado  Cambiar

Datos empleado

Nombre:Cesar

Edad:45. Es mayor!!!



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Código completo

- Archivo **empleado.component.html**

```
<!--Cuadro de texto de tipo numero para la edad-->
<p>Ingrese la edad del empleado
  <input type="number" name="" id="" #txtEdad>
  <button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>
</p>
```

```
<p class="etiqueta">Datos empleado</p>
<p class="etiqueta">Nombre:{{getNombre()}}</p>
<p class="etiqueta">Edad:<span id="lblEdad"></span></p>
<p class="etiqueta">Profesión:{{getProfesion()}}</p>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 Interpolación – Código completo

- Archivo **empleado.component.ts**

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

@Component({
  selector: 'app-empleado',
  templateUrl: './empleado.component.html',
  styleUrls: ['./empleado.component.css']
})
export class EmpleadoComponent implements OnInit {
  //Agregamos atributos a nuestra clase
  private vcapellido:string="Flores";
  private vcnombre:string="Cesar";
  private vcedad:number=40;
  private vcprofesion:string="Programador";

  constructor() { }
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

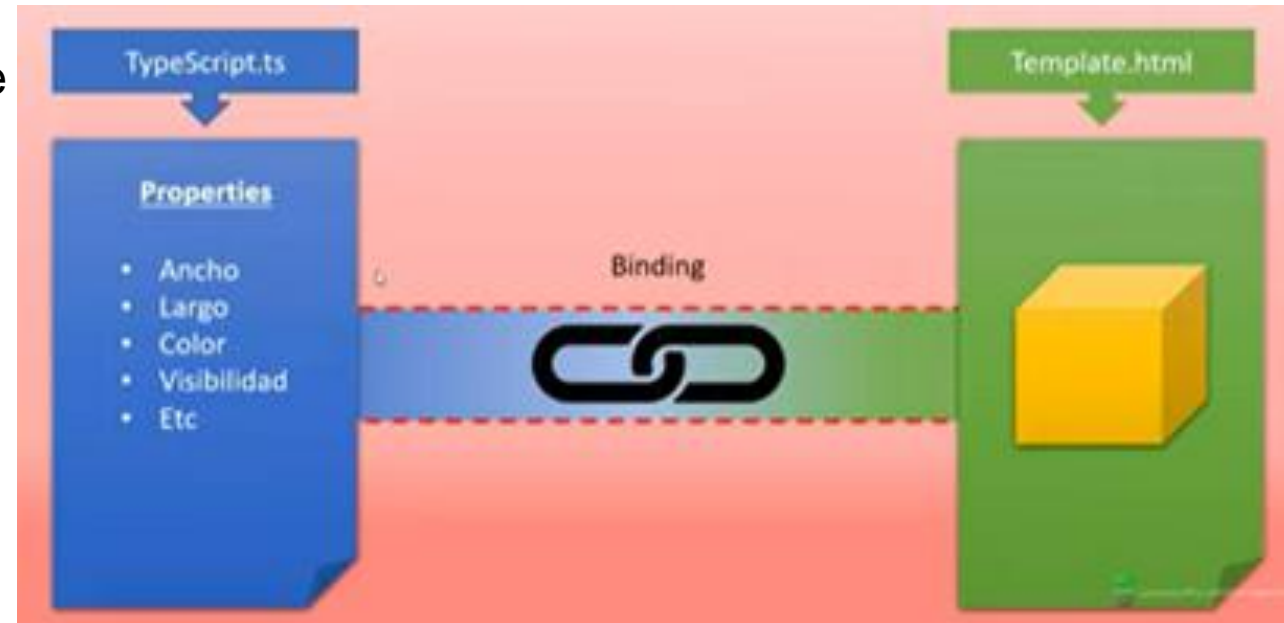
## Property Binding



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Property Binding

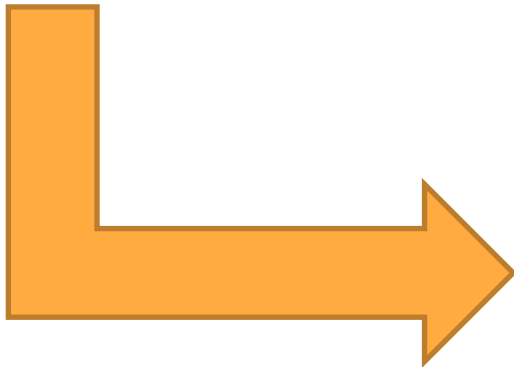
- Binding = «Unión», «Vínculo», «Puente»
- Es la acción de unir las propiedades que nosotros declaramos o se tiene de un objeto en el archivo ts con el objeto que se inserta o incluye en el archivo html o template.
- El motivo de esto es poder modificar dinámicamente el comportamiento de esta a lo largo de la vida de nuestra App.
- La lectura de un dato de una DB/s y lleve al cambio de estado de un control en función del dato obtenido.



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Uso Property Binding

- Sobre el cuadro de texto edad de nuestro html podemos manipular su propiedad de habilitado o deshabilitado mediante Property Binding, haciendo lo siguiente:
- En la clase empleado que se encuentra en el archivo **empleado.component.ts** agregamos un atributo publico que maneje el estado de habilitación del objeto text.



```
TS empleado.component.ts U x
miprimeraapp > src > app > empleado > TS empleado.component.
7    })
8    export class EmpleadoComponent implements OnInit {
9        //Agregamos atributos a nuestra clase
10       private vcapellido:string="Flores";
11       private vcnombre:string="Cesar";
12       private vcedad:number=40;
13       private vcprofesion:string="Programador";
14       private vestadocheck:boolean=false;
15    }
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Uso Property Binding

- En el archivo template **empleado.component.html** del componente empleado realizamos el siguiente agregado de la propiedad que determina el estado del objeto text para permitir el ingreso o no de datos, tradicionalmente se indicaría mediante la propiedad **disabled** no permite cargar dato y sin **disable** permite cargar datos:

```
>Ingrese la edad del empleado  
<input type="number" name="" id="" #txtEdad>
```

Ingrese la edad del empleado

Ingrese la edad del empleado

 Cambiar

```
>Ingrese la edad del empleado  
<input type="number" name="" id="" #txtEdad disabled>  
<button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>
```

Ingrese la edad del empleado

 Cambiar

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Uso Property Binding

- Aplicando Property Binding y un método **getEstadoCheck** quien me devuelvo el estado del atributo **vestadoocheck** quedaría dicha propiedad de la siguiente manera:

```
<input type="number" name="" id="" #txtEdad [disabled]=getEstadoCheck()>
```

Dentro de empleados

```
private vcprofesion:string= Programador  
private vestadoocheck:boolean=false;
```

Ingrese la edad del empleado

```
<p>Ingrese la edad del empleado  
<input type="number" name="" id="" #txtEdad [disabled]=getEstadoCheck()>  
<button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>
```

Dentro de empleados

```
private vcprofesion:string= Programador  
private vestadoocheck:boolean=true;
```

Ingrese la edad del empleado

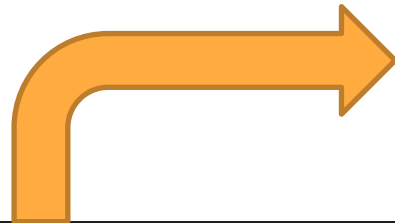
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Event Binding

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo - Uso Event Binding

- Aplicando Event Binding mediante un control checkbox para que al habilitarlo se habilite el control text y al deshabilitarlo el check se deshabilite el control que permite la carga de la edad.



Dentro de empleados

Ingrese la edad del empleado   ☐ Habilitar text

```
<p>Ingrese la edad del empleado  
  <input type="number" name="" id="" #txtEdad [disabled]=getEstadoCheck()>  
  <button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>  
  <input type="checkbox" name="" id="chkHabilitar" (change) ="ControlarCheckBox($event)">Habilitar text  
</p>
```



Dentro de empleados

Ingrese la edad del empleado   ☒ Habilitar text

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo - Uso Event Binding

- Dentro del objeto empleado existe un atributo sobre el cual se guarda los estados del control check aparte existe un método que recibe el **event** del **objeto check** y dentro de esta recuperaremos su estado y se lo pasara al método que carga al correspondiente atributo que habilitar o no el control text

```
private vcprofesion:string="Programador";  
private vcestadocheck:boolean=false;
```

```
public ControlarCheckBox(eventoctrlcheck: Event):boolean  
{//Función que permite habilitar el control text  
  //Recupero estado del check  
  this.setEstadoCheck((<HTMLInputElement>eventoctrlcheck.target).checked);  
  
  return this.getEstadoCheck();  
}
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo - Uso Event Binding – Código completo – empleado.component.html

```
<!--Cuadro de texto de tipo numero para la edad-->
```

```
<p>Ingrese la edad del empleado
```

```
  <input type="number" name="" id="" #txtEdad [disabled]=getEstadoCheck()>
```

```
  <button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>
```

```
  <input type="checkbox" name="" id="chkHabilitar" (change) ="ControlarCheckBox($event)">Habilitar text
```

```
</p>
```

```
<p class="etiqueta">Datos empleado</p>
```

```
<p class="etiqueta">Nombre:{{getNombre()}}</p>
```

```
<p class="etiqueta">Edad:<span id="lblEdad"></span></p>
```

```
<p class="etiqueta">Profesión:{{getProfesion()}}</p>
```



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 - Uso Property Binding – Código completo – empleado.component.ts

```
export class EmpleadoComponent implements OnInit {  
  //Agregamos atributos a nuestra clase  
  private vcapellido:string="Flores";  
  private vcnombre:string="Cesar";  
  private vcedad:number=40;  
  private vcprofesion:string="Programador";  
  private vcestadocheck:boolean=false;  
  
  //Métodos  
  public getEdad() { return this.vcedad <18 ? this.vcedad+". Es menor!!!":this.vcedad+". Es mayor!!!"; }  
  public getEstadoCheck() { return this.vcestadocheck; }  
  public setEstadoCheck(dato:boolean) { this.vcestadocheck=dato; }
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejemplo 2 - Uso Property Binding – Código completo – empleado.component.ts

```
public AnalizarEdad(dato:string) {  
    //El dato que viene como parametro se lo paso a método setEdad para  
    //que cargue el atributo vcedad, se lo convierte a número  
    this.setEdad(parseInt(dato));  
    //Se muestra el resultado del método getEdad sobre el control span  
    (<HTMLInputElement>document.getElementById("lblEdad")).innerText=this.getEdad();  
}  
  
public ControlarCheckBox(eventoctrlcheck: Event):boolean  
{//Función que permite habilitar el control text  
    //Recupero estado del check  
    this.setEstadoCheck((<HTMLInputElement>eventoctrlcheck.target).checked);  
  
    return this.getEstadoCheck();  
}
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## **Event Binding – Listado de eventos y su uso**

A continuación se deja un link para que puedan observar los distintos tipos de eventos que existen en Angular y como se los puede utilizar

[http://www.w3bai.com/es/angular/angular\\_events.html](http://www.w3bai.com/es/angular/angular_events.html)

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

- El binding bidireccional nos sirve para poder tener un flujo de información tanto desde el Template hacia el archivo TypeScript como del TypeScript al Template.
- Ejemplo, agregar un cuadro de texto y una etiqueta, se ingresara el deporte preferido del empleado en el control text y este se deberá de ir mostrando en simultaneo sobre el etiqueta.
- El código para generar este diseño en el archivo **empleado.component.html** es el siguiente:

```
<p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte></p>  
<p class="etiqueta">Deporte preferido ingresado: {{vcdeporte}}</p>
```



Ingrese deporte preferido:

Deporte preferido ingresado: Ninguno

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

- Dentro de nuestro archivo **empleado.component.ts** declamáramos una atributo con alcance global y le definimos un valor inicial:

```
//Atributo con alcance público  
vcdeporte:string="Ninguno";
```

- Dicho valores reflejado en la etiqueta de nuestra página ya que se esta haciendo un binding entre el TypeScript => Template (Binding por propiedad)

```
//Atributo con alcance público  
vcdeporte:string="Ninguno";  
  
constructor() { }
```

Ingrese deporte preferido:

Deporte preferido ingresado: Ninguno

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

- Agregamos a nuestro control text un evento que al escribir en dicho control la información se deberá de ir mostrando sobre la etiqueta, el código de nuestro archivo **empleado.component.html** queda de la siguiente manera:

```
<p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte  
(input)="ReflejarContenido($event)"></p>
```

```
<p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte (input)="ReflejarContenido($event)"></p>  
<p class="etiqueta">Deporte preferido ingresado: {{vcdeporte}}</p>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

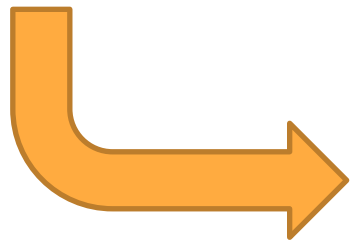
- Luego se deberá de realizar la creación del evento al que se esta invocando desde el input text, teniendo como argumento los evento asociado a dicho objeto, dicha programación se realiza sobre el archivo **empleado.component.ts**.
- **El valor que se escribe en el input text llega a nuestra atributo por medio de la propiedad value que trae asociado los event del objeto input es decir desde el Template => TypeScript (Binding por event)**



# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

```
public ReflejarContenido(dato:Event){ this.vcdeporte=(<HTMLInputElement>dato.target).value; }
```



```
TS empleado.component.ts U X
miprimeraapp > src > app > empleado > TS empleado.component.ts > EmpleadoComponent
15
16 //Atributo con alcance público
17 vcdeporte:string="Ninguno";
18
19 //Método que permite reflejar el contenido en nuestro atributo
20 public ReflejarContenido(dato:Event){
21 |   this.vcdeporte=(<HTMLInputElement>dato.target).value;
22 | }
23
24
```

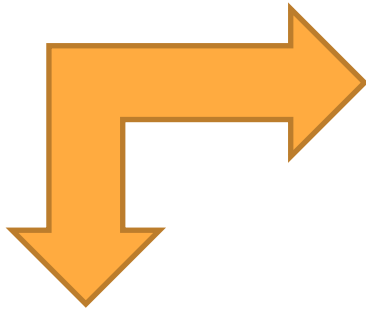
# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

- Pero cada vez que se refresca la página de nuestra App el text input y etiqueta vuelven a su estado inicial, el text queda vacío y la etiqueta muestra el cartel de ninguno.
- Para evitar esto se aplica un binding bidireccional entre el **Template => TypeScript y viceversa**.
- Para ello realizamos los siguientes cambios, sobre el archivo **empleado.component.ts**, se borra la función que lleva adelante el proceso de reflejar la información del text a la etiqueta y luego borramos su llamada en el archivo **empleado.component.html**, nos queda este código como sigue a continuación, simularemos dicha eliminación comentando las respectivas líneas de códigos.
- A esto se lo conoce como uso de **Banana in Box**.

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional



```
TS empleado.component.ts U X  <> empleado.component.html 1, U
miprimeraapp > src > app > empleado > TS empleado.component.ts > EmpleadoComponent

16 //Atributo con alcance público
17 vcdeporte:string="Voley";
18
19 /*
20 //Método que permite reflejar el contenido en nuestro atributo
21 public ReflejarContenido(dato:Event){
22   this.vcdeporte=(<HTMLInputElement>dato.target).value;
23 }
24 */
25
```

```
<> empleado.component.html U X
miprimeraapp > src > app > empleado > <> empleado.component.html > ...

12
13 <!--
14 <p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte (input)="ReflejarContenido($event)"></p>
15 -->
16 <p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte ></p>
17
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional

- A continuación se realiza la siguiente modificación sobre el código del archivo **empleado.component.html**, quedando el siguiente código dentro del control input text.

<p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte [(ngModel)]=vcdeporte"></p>

- Luego deberemos de importar la librería de Angular para que nos reconozca las funcionalidades de **ngModel**, para ello realizamos lo siguiente, nos dirigimos al archivo **app.module.ts** de nuestra App y dentro de este archivo agregamos la siguiente línea de código.

```
...
imports: [
  BrowserModule, FormsModule
],...
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Binding Bidireccional – Código completo - empleado.component.html.

```
<!--Cuadro de texto de tipo numero para la edad-->
<p>Ingrese la edad del empleado
  <input type="number" name="" id="" #txtEdad [disabled]=getEstadoCheck()>
  <button (click)="AnalizarEdad(txtEdad.value)">Cambiar</button>
  <input type="checkbox" name="" id="chkHabilitar" (change) ="ControlarCheckBox($event)">Habilitar text
</p>

<p class="etiqueta">Datos empleado</p>
<p class="etiqueta">Nombre:{{getNombre()}}</p>
<p class="etiqueta">Edad:<span id="lblEdad"></span></p>
<p class="etiqueta">Profesión:{{getProfesion()}}</p>

<p>Ingrese deporte preferido: <input type="text" name="" id="" #txtDeporte [(ngModel)]=vcdeporte"></p>

<p class="etiqueta">Deporte preferido ingresado: {{vcdeporte}}</p>
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

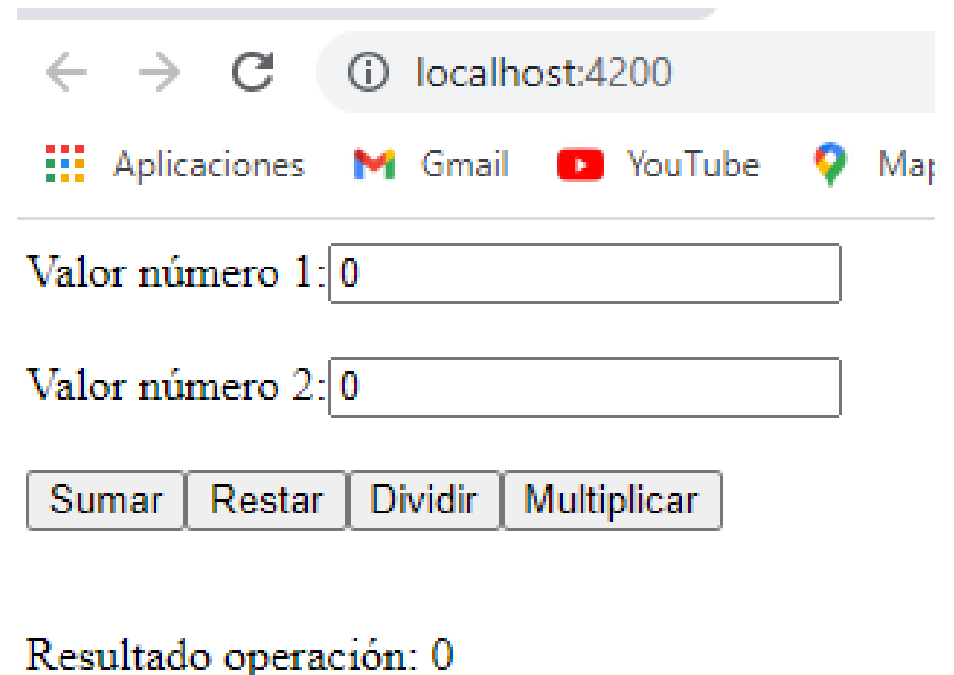
- **Archivo empleado.component.ts**
  - Se elimino la función **ReflejarContenido(dato:Event) {}**.
  - Se deja el atributo **vcdeporte:string="Voley"**;
- **Archivo app.module.ts** – Lugar en donde se agrega la referencia a librería Angular

```
@NgModule({  
  declarations: [  
    AppComponent, EmpleadosComponent, EmpleadoComponent  
  ],  
  imports: [  
    BrowserModule, FormsModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})
```

# Módulo 4 - Front End - Desarrollo Web Dinámico

## Ejercicio

- Genere una calculadora que permita realizar las 4 operaciones básicas con Angular, sumar, restar, dividir y multiplicar.
- Deberá de utilizar todos los temas tratados hasta ahora.
- Vista previa de nuestra pagina web calculadora básica.
- Las funciones que invocara cada botón se llamen
  - Suma()
  - Resta()
  - Multiplicacion()
  - Division()
- Todas las funciones recibirán los valores a trabajar como parámetros de la función.



A screenshot of a web browser displaying a basic calculator application. The browser's address bar shows 'localhost:4200'. Below the address bar, there are links for 'Aplicaciones', 'Gmail', 'YouTube', and 'Map'. The calculator interface includes two input fields labeled 'Valor número 1:' and 'Valor número 2:', both containing the value '0'. Below these fields are four buttons labeled 'Sumar', 'Restar', 'Dividir', and 'Multiplicar'. At the bottom, there is a label 'Resultado operación:' followed by the value '0'.

# Fin Presentación.

---