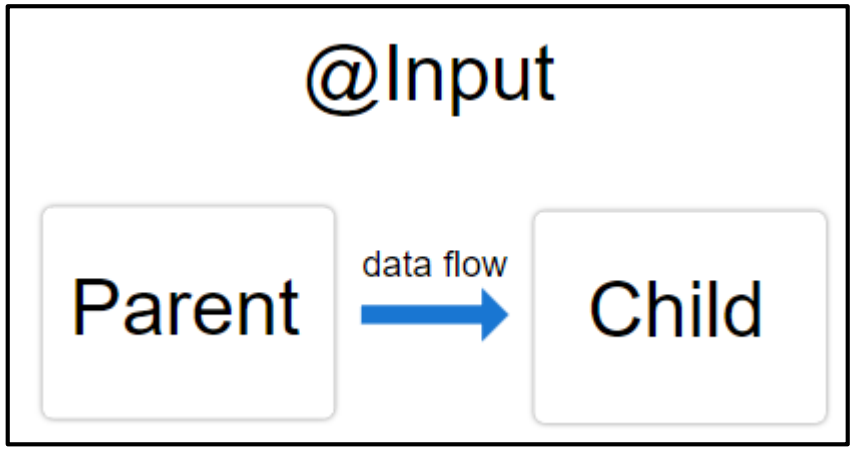
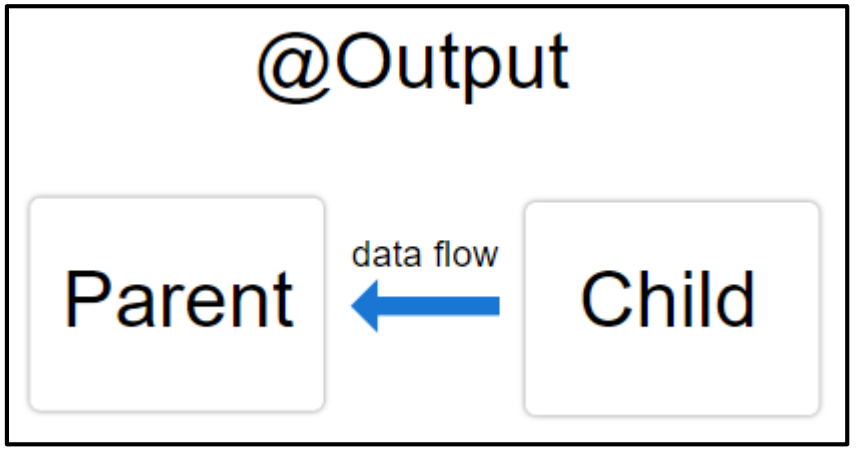
5.1. Directiva @Input

1. Comunicación entre componentes

- La comunicación entre un componente padre y un hijo se realiza a través de las directiva @Input()
- La directiva @Output() permite enviar información del componente hijo al padre.





2. Ejemplos de uso

- Vamos a partir de un ejemplo sobre un componente individual.
- En el fichero TS tenemos dos estructuras de datos, llamadas datos1 y datos2 con estructura similar pero con datos distintos.

2. Ejemplos de uso

```
export class AppComponent {
  title = 'nohijos';

  datos1 = [
    {
      nombre: 'Pepe',
      apellido: 'Perez',
      edad: 25,
      sexo: 'M'
    },
    {
      nombre: 'Juan',
      apellido: 'Perez',
      edad: 25,
      sexo: 'M'
    }];
```

```
  datos2 = [
    {
      nombre: 'Pepe2',
      apellido: 'Perez2',
      edad: 25,
      sexo: 'M'
    },
    {
      nombre: 'Juan2',
      apellido: 'Perez2',
      edad: 25,
      sexo: 'M'
    }];

}
```

2. Ejemplos de uso

- En el template (HTML) vamos a mostrar dichos valores mediante una lista
- Para ello debemos recorrer ambos array y mostrarlos mediante un ngFor

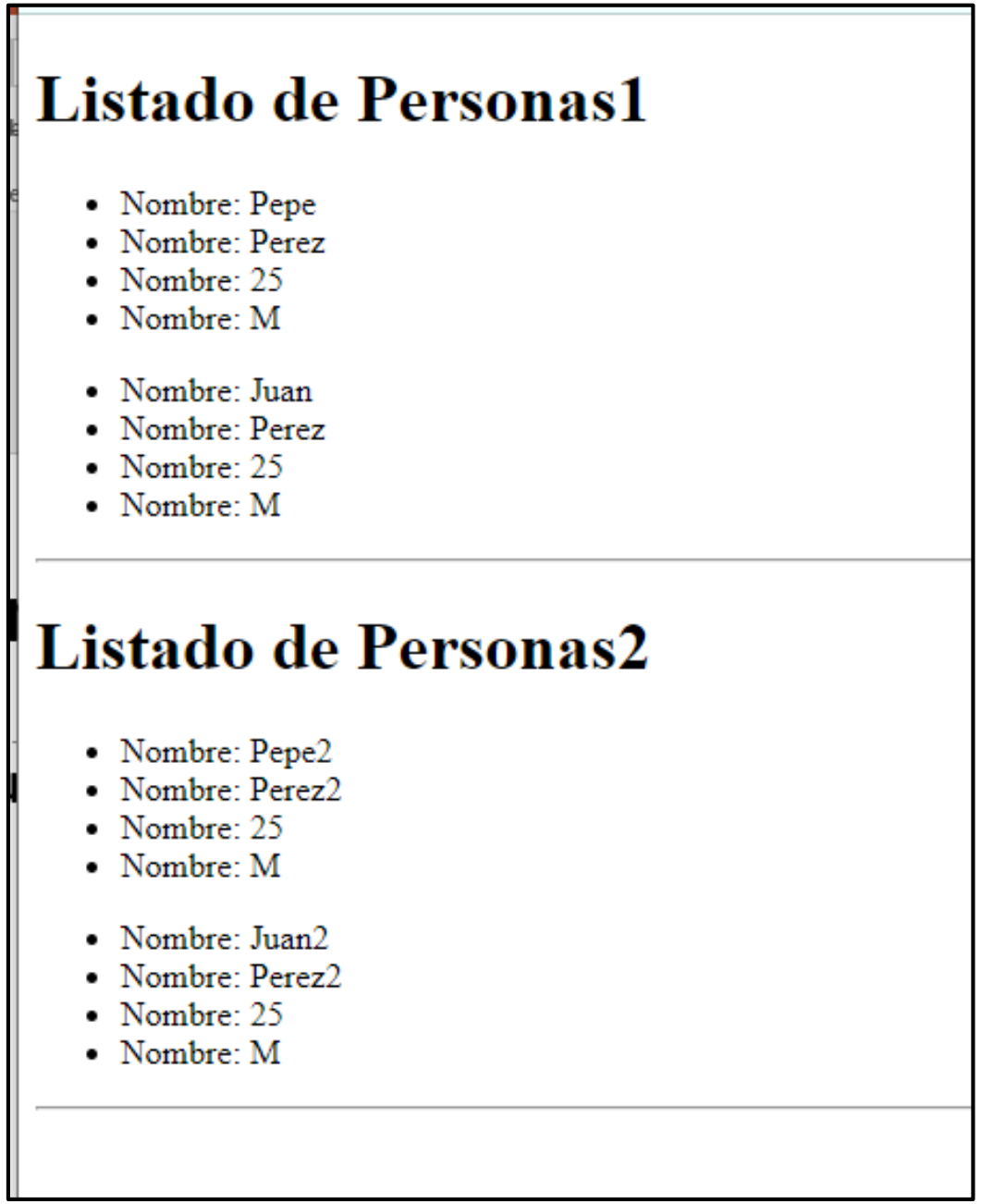
2. Ejemplos de uso

```
<h1>Listado de Personas1</h1>
<ul *ngFor="let elemento of datos1">
  <li>Nombre: {{elemento.nombre}}</li>
  <li>Nombre: {{elemento.apellido}}</li>
  <li>Nombre: {{elemento.edad}}</li>
  <li>Nombre: {{elemento.sexo}}</li>

</ul>
<hr>
```

```
<h1>Listado de Personas2</h1>
<ul *ngFor="let elemento of datos2">
  <li>Nombre: {{elemento.nombre}}</li>
  <li>Nombre: {{elemento.apellido}}</li>
  <li>Nombre: {{elemento.edad}}</li>
  <li>Nombre: {{elemento.sexo}}</li>

</ul>
<hr>
```

2. Ejemplos de uso

- El resultado obtenido es el siguiente:



Listado de Personas1

- Nombre: Pepe
- Nombre: Perez
- Nombre: 25
- Nombre: M

- Nombre: Juan
- Nombre: Perez
- Nombre: 25
- Nombre: M

Listado de Personas2

- Nombre: Pepe2
- Nombre: Perez2
- Nombre: 25
- Nombre: M

- Nombre: Juan2
- Nombre: Perez2
- Nombre: 25
- Nombre: M

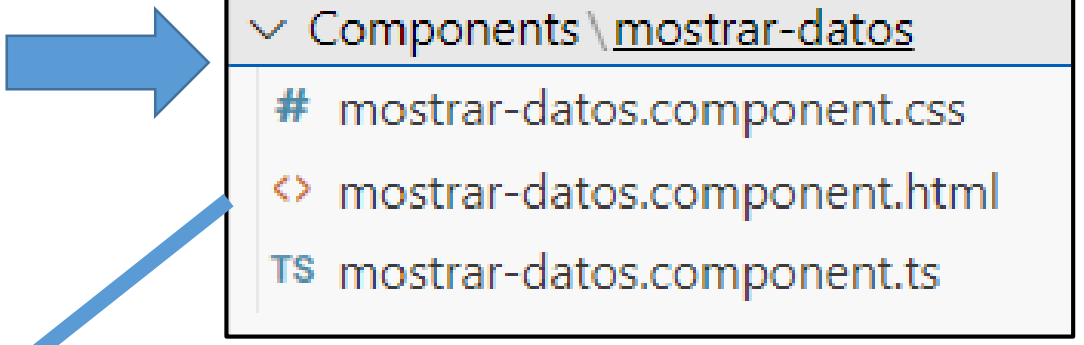
---

3. División de la aplicación

- Vamos a dividir la aplicación en varios componentes (que es lo habitual en Angular)
- Tendremos un componente padre y un hijo. Vamos a ver como se realiza la comunicación entre dichos componentes.
- Desde la terminal añadimos un componte nuevo a nuestra aplicación llamado **mostrarDatos**.
  - ***ng g c mostrarDatos***

3. División de la aplicación

```
>ng g c Components\mostrarDatos
CREATE src/app/Components/mostrar-datos/mostrar-datos.component.html (29 bytes)
CREATE src/app/Components/mostrar-datos/mostrar-datos.component.spec.ts (667 bytes)
CREATE src/app/Components/mostrar-datos/mostrar-datos.component.ts (236 bytes)
CREATE src/app/Components/mostrar-datos/mostrar-datos.component.css (0 bytes)
UPDATE src/app/app.module.ts (512 bytes)
```



3. División de la aplicación

- Vamos a dividir ahora el contenido original entre estos dos componentes. En el template original teníamos dos <ul> con dos *ngFor recorriendo las estructuras.
- Ahora lo que vamos a hacer es que el componente **mostrarDatos** reciba del padre el array de datos y los muestre por pantalla (@Input).

3. División de la aplicación

```
export class MostrarDatosComponent {
@Input() datos;
```

```
@Component({
    El miembro 'datos' tiene un tipo 'any' implícitamente. ts(7008)
    (property) MostrarDatosComponent.datos: any
})  Ver el problema (Alt+F8)    Corrección Rápida (Ctrl+.)
ex
@Input() datos;
```

3. División de la aplicación

- Nos dará error de **typescript**. Para solucionar este error debemos añadir esta línea en el fichero **tsconfig.json**

```
"compilerOptions": {
  "outDir": "./dist/out-tsc",
  "forceConsistentCasingInFileNames": true,
  "strict": false,
  "noImplicitOverride": true,
```

3. División de la aplicación

- La sección donde mostramos los datos la vamos a pasar al componente hijo.

mostrar-datos.component.html

src > app > Components > mostrar-datos > mostrar-datos.com

Go to component

```
1  <ul *ngFor="let elemento of datos">
2      <li>Nombre: {{elemento.nombre}}</li>
3      <li>Nombre: {{elemento.apellido}}</li>
4      <li>Nombre: {{elemento.edad}}</li>
5      <li>Nombre: {{elemento.sexo}}</li>
6
7    </ul>
8
9  <router-outlet></router-outlet>
```

3. División de la aplicación

- Solo nos queda pasar los datos a mostrar desde el componente padre al hijo
- Hay que tener en cuenta que la variable a utilizar debe ser la misma que la utilizada en @Input

3. División de la aplicación

app.component.html

src > app > app.component.html > ...

Go to component

```
1  <h1>Listado de Personas1</h1>
2  <app-mostrar-datos [datos]="datos1"></app-mostrar-datos>
3  <hr />
4
5  <h1>Listado de Personas2</h1>
6  <app-mostrar-datos [datos]="datos2"></app-mostrar-datos>
7  <hr />
8
9  <router-outlet></router-outlet>
10
```

```
export class MostrarDatosComponent {
@Input() datos;
```

3. División de la aplicación

- El resultado es el mismo que en la versión anterior (con un solo componente)