UA . MASTER MOVILES

MÁSTER UNIVERSITARIO EN DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

PROGRAMACIÓN HIPERMEDIA PARA DISPOSITIVOS MÓVILES

Ionic v4 – Binding, Directivas y Servicios

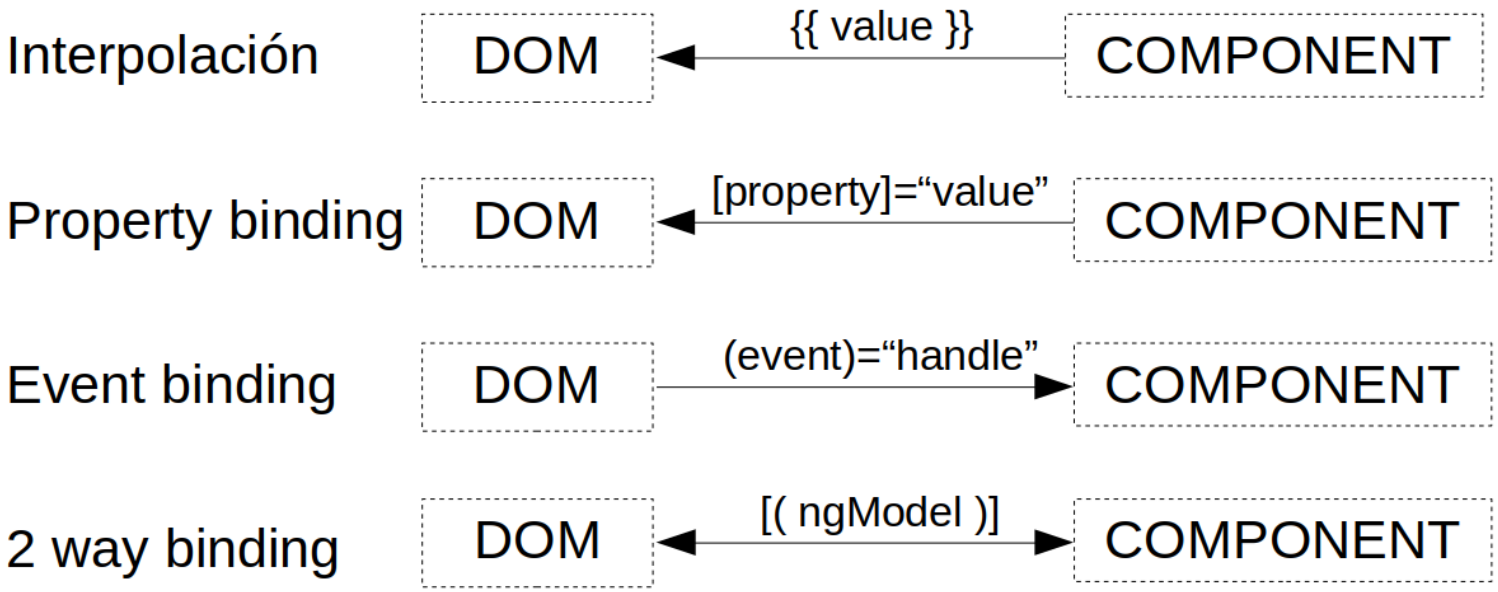
CONTENIDO

1. Binding
    1. Interpolación
    2. Variables
    3. Bidireccional
2. Directivas
    1. Directivas estructurales
3. Servicios o proveedores de contenidos

BINDING

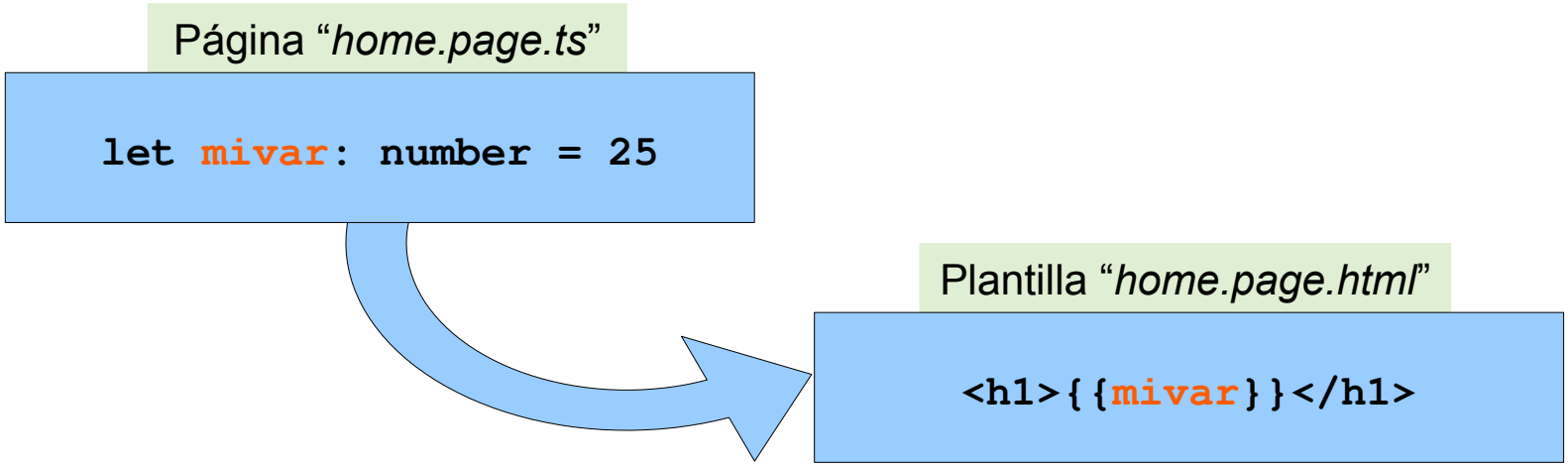
BINDING

- El binding nos permite comunicar datos entre las páginas o componentes y las plantillas.
- Podemos utilizar cuatro tipos de Binding:



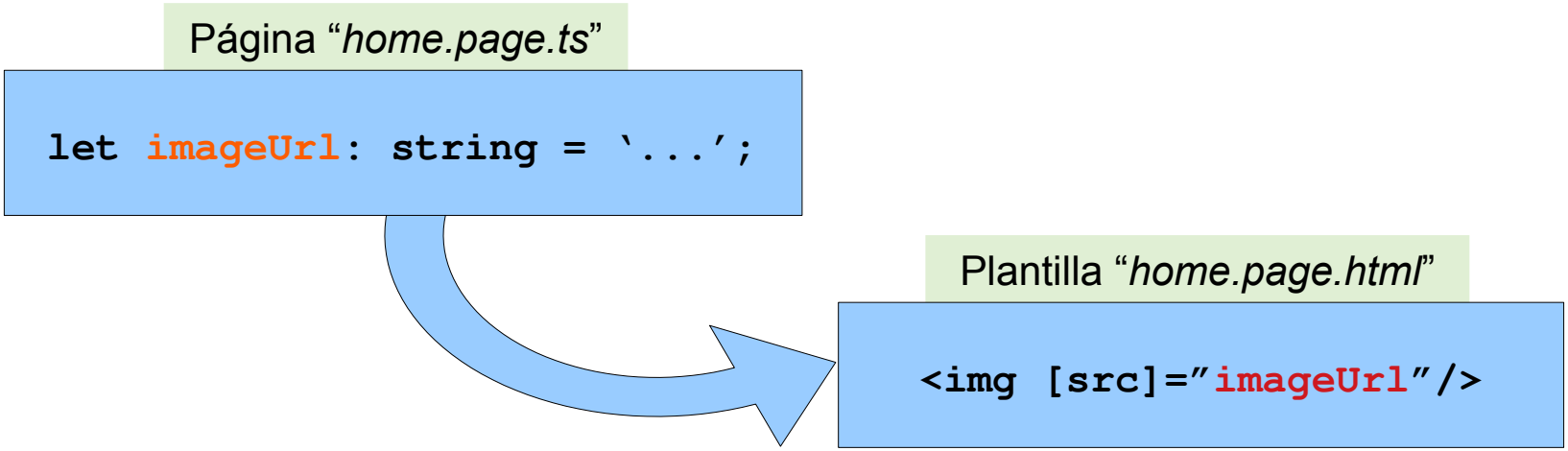
BINDING DE VARIABLES

- El binding por **interpolación** nos permite mostrar datos del componente en la plantilla.
- Simplemente tenemos que indicar en la plantilla el mismo nombre de variable entre llaves dobles.



BINDING DE PROPIEDADES

- El binding de **propiedades** dar valor a propiedades de una etiqueta.
- En este caso en la plantilla tenemos que indicar el nombre de la propiedad a definir entre llaves y asignarle como valor el nombre de la variable a utilizar:



BINDING DE VARIABLES Y PROPIEDADES

- En la plantilla (`home.page.html`):

```
<p><strong>Nombre:</strong>{{nombre}}</p>
<p><strong>Edad:</strong>{{edad}}</p>
<p><strong>Dirección:</strong>{{dir.calle +' '+ dir.num}}</p>
<img [src]="logoUrl">
<button [disabled]="isDisabled">Púlsame</button>
```

- En la clase de la página asociada (`home.page.ts`):

```
export class HomePage {
  nombre: string = 'Pedro';
  edad: number = 25;
  dir: {calle: string, num: number};
  logoUrl: string = '/assets/logo.png';
  isDisabled: boolean = true;
  constructor() {
    this.dir = {calle: 'C/calle', num: 16};
  }
}
```

BINDING DE EVENTOS

- Podemos asociar cualquier evento definido en HTML (https://www.w3schools.com/jsref/dom_obj_event.asp) para que ejecuten un método de la clase.
- En la plantilla tendremos que poner el nombre del evento entre paréntesis, quitando el prefijo "on" del nombre HTML:

  `onclick, onblur, onchange → click, blur, change`

- Para responder a la pulsación de botón, en la plantilla pondremos:

```
<ion-button (click)="botonPulsado()">Púlsame!</ion-button>
```

- Y en la clase asociada:

```
export class HomePage {
  constructor() { }

  botonPulsado() {
     console.log('Se ha pulsado el botón!');
  }
}
```

BINDING BIDIRECCIONAL

- El binding bidireccional comunica los valores en ambas direcciones:

  → El valor asignado a la variable en el controlador (`.ts`) se enviará a la vista.

  ← Los cambios realizados en la vista se almacenarán en el controlador.

- Normalmente se utiliza en campos tipo *input* usando la notación:
  `[(ngModel)] = "variable"`
- En la plantilla tendríamos que poner:

```
<ion-label fixed>Nombre</ion-label>
<ion-input [(ngModel)]="nombre" name="nombre" type="text"></ion-input>
```

- En la clase asociada:

```
export class HomePage {
  nombre: string = 'Pedro';

  constructor() { }
}
```

DIRECTIVAS

DIRECTIVAS

- Permiten definir pedazos de código reutilizables dentro de un proyecto.
- Con Angular podemos definir tres tipos de directivas:
  - Componentes (o páginas).
  - Directivas para atributos.
  - Directivas estructurales.
- Las directivas para atributos modifican la apariencia o comportamiento:

```
<ion-input clearInput value="Bórrame"></ion-input>

<ion-item detail>...</ion-item>
```

DIRECTIVAS ESTRUCTURALES

- Modifican la estructura de la página añadiendo o quitando elementos.
- Por ejemplo, podemos añadir condiciones tipo “if” y bucles “for” a la plantilla de la forma:
- **`*ngIf`**

```
<div *ngIf=isShown>
    <h1>Welcome!</h1>
</div>
```

En la clase `.ts` tendríamos que definir la variable “**`isShown`**” de tipo booleano

- **`*ngFor`**

```
<ul>
    <li *ngFor="let item of listItems">
        {{ item }}
    </li>
</ul>
```

En la clase `.ts` tendríamos que definir la variable “**`listItems`**” de tipo array

DIRECTIVAS ESTRUCTURALES: EJEMPLO

- En la plantilla:

```
<ion-button (click)="toggleWelcome()">Toggle Welcome</ion-button>
<div *ngIf=isShown>
    <li *ngFor="let item of listItems">
        {{ item }}
    </li>
</div>
```

- En la clase asociada:

```
export class HomePage {
    isShown: boolean = false;
    listItems: number[] = [1, 2, 3, 4];

    constructor() {}

    toggleWelcome() {
        this.isShown = !this.isShown;
    }
}
```

SERVICIOS O PROVEEDORES DE CONTENIDOS

PROVEEDORES DE CONTENIDOS O SERVICIOS

- Los proveedores de contenidos o servicios se encargan de proporcionar los datos obtenidos de forma local o remota.
- Se instancia utilizando el patrón Singleton.
- Para crear un proveedor:

```
$ ionic g service services/datos
```

- Esto nos creará el servicio `DatosService` en:

`src/app/services/datos.service.ts`

PROVEEDORES DE CONTENIDOS O SERVICIOS

- El fichero con el servicio generado contendrá por defecto el siguiente código:

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class DatosService {
  constructor() { }
}
```

- En esta clase tenemos podemos añadir todos los métodos y propiedades que queramos permitir el acceso a los datos proporcionados por la misma.
- La clase puede almacenar u obtener los datos como queramos: datos fijos en la propia clase, accediendo a almacenamiento local, accediendo a Internet, etc.

PROVEEDORES DE CONTENIDOS: EJEMPLO

- Creamos un servicio con datos estáticos:

```
export class DatosService {
  listItems: string[] = ['One', 'Two', 'Three', 'Four'];
  constructor() {}
  getItems(): string[] {
    return this.listItems;
  }
}
```

- Y para utilizarlo desde una página:

```
import { DatosService } from '../../services/datos.service';

@Component({…})
export class Pagina2Page {
  listItems: string[];
  constructor(private datos: DatosService) {
    this.listItems = datos.getItems();
  }
}
```

¿PREGUNTAS?