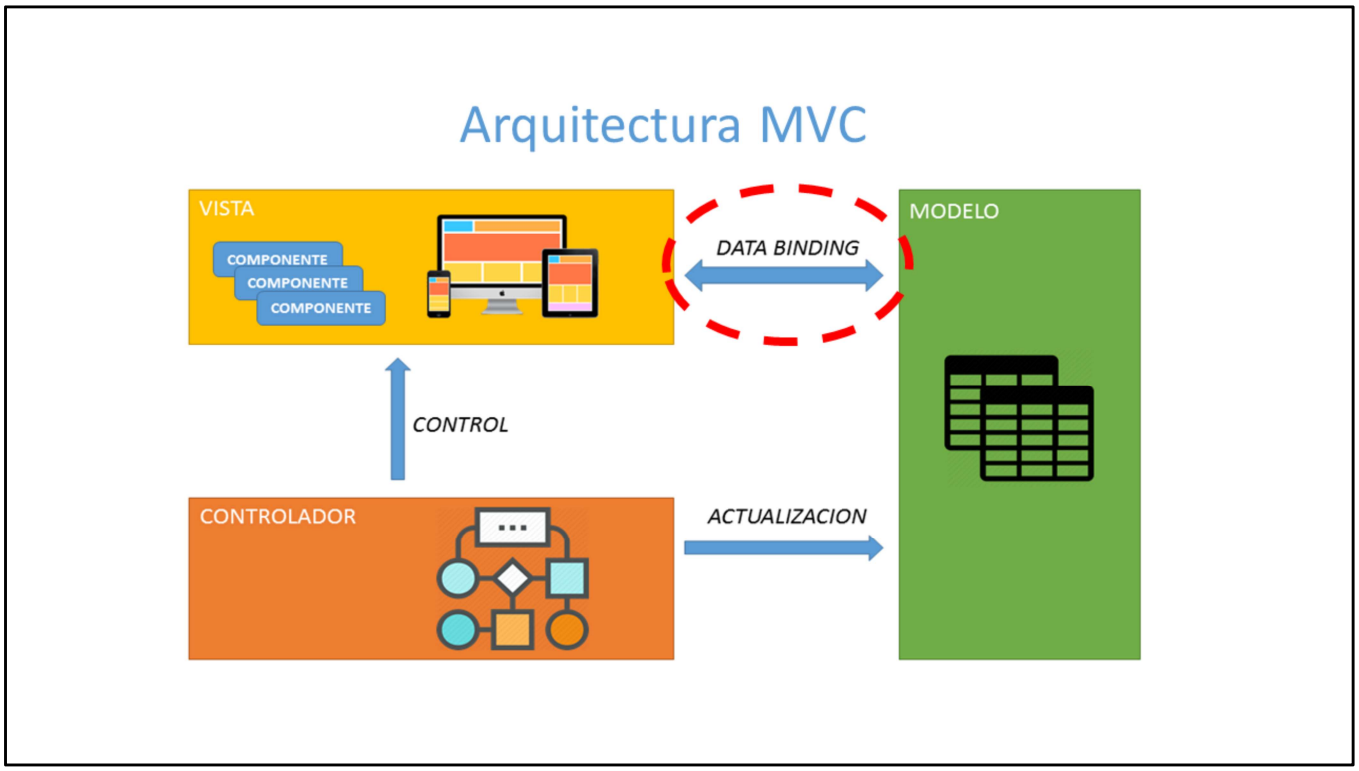
Tema 4 – Mostrar contenido dinámico

- Interpolación
- Formatear datos con pipes
- Binding de propiedades
- Binding de clases y estilos CSS
- Directivas estructurales
- Binding de eventos (manejo de eventos)

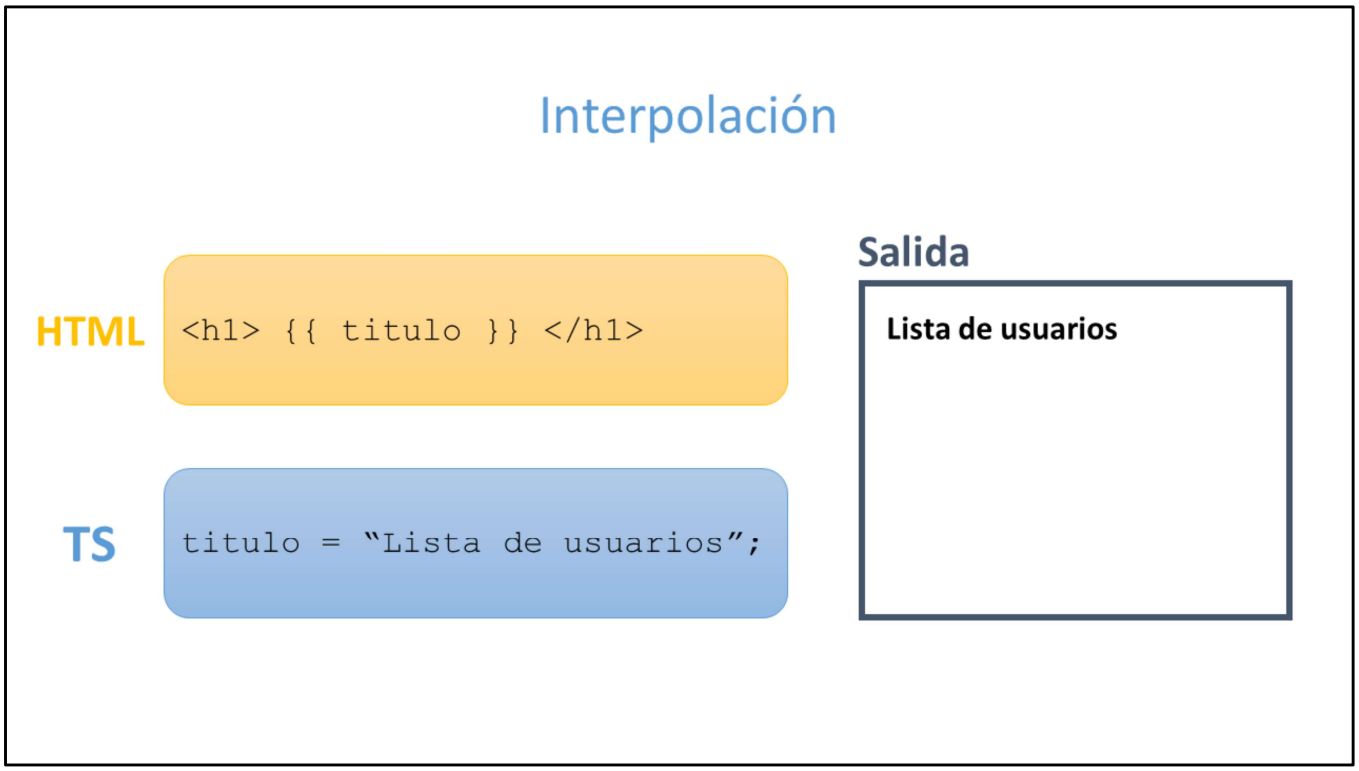
En este tema vamos a conocer los mecanismos que usa Angular para que el contenido visual de los componentes sea dinámico, lo cual es fundamental en un frontend web.

Como se puede ver ya simplemente en el índice, el concepto de “binding” se usa en la mayoría de estos mecanismos.



Si recordamos, hablamos ya del “data binding” cuando vimos las relaciones entre las capas “Vista” y “Modelo” de la arquitectura MVC.

Decíamos que el binding era la capacidad para enlazar datos del modelo con componentes visuales, y viceversa. De esa forma al modificar el modelo se actualiza la vista y al cambiar la vista se actualiza el modelo.



Vimos anteriormente uno de los métodos para mostrar contenido dinámico en nuestro componente: la interpolación, mediante expresiones javascript entre dobles llaves {{}}, insertadas directamente en el código HTML.

No es necesario que sea un nombre de propiedad. Puede ser cualquier cosa que devuelva un valor, como expresiones o funciones.

Cuando se trata de campos del componente, no es necesario usar "this" delante, va implícito.

Ejemplos:

```
<h3>Interpolacion</h3>
<div>{{ titulo }}</div>
<div>{{ titulo.toUpperCase() }}</div>
<div>{{ titulo + ' ' + titulo2 }}</div>
```

Formatear datos con pipes

HTML

```
<p> {{ titulo | uppercase }} </p>

<p> {{ fecha | date:'dd/MM/yyyy' }} </p>
```

uppercase | lowercase | titlecase | date | decimal | currency

https://angular.io/api?type=pipe

En una vista es fundamental el poder controlar cómo se presentan los datos.

Angular permite formatear los datos a mostrar mediante el operador pipe ("|") en las interpolaciones

En el ejemplo, estamos formateando la expresion "titulo" con el pipe "uppercase", que convierte el texto a mayusculas.

Existen muchos pipes prefabricados en Angular (https://angular.io/api?type=pipe), tanto para texto como para fechas, numeros, etc.

Los más comunes: uppercase, lowecase, titlecase, date, decimal, currency

Binding de propiedades

HTML

```
<img src="{{ urlImagen }}">
<img [src]="urlImagen" ✔
```

TS

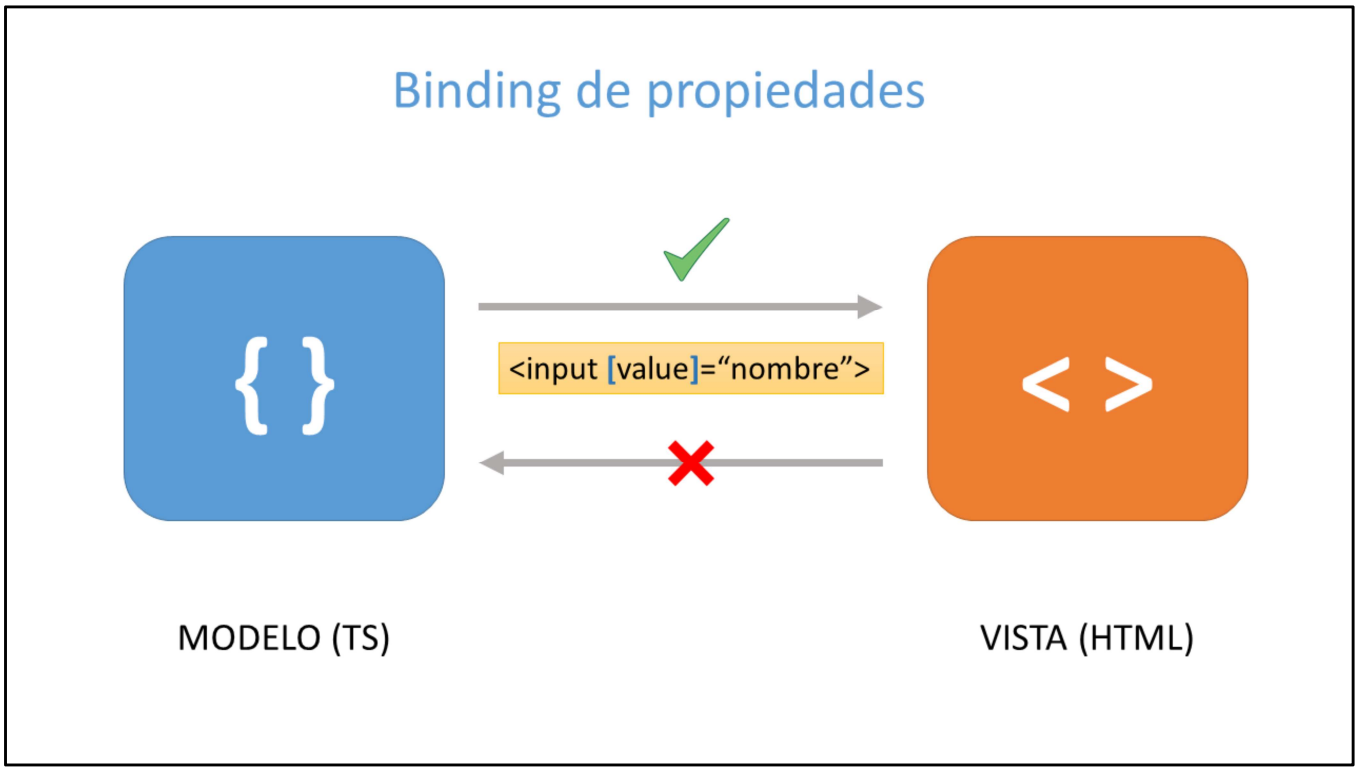
```
urlImagen = "img/perro.png";
```

Salida

La interpolación se puede usar para sustituir cualquier contenido en la plantilla HTML. Suponiendo que tenemos un campo en nuestro componente que especifica la URL para una imagen, llamado urlImagen, podríamos usarlo en la plantilla HTML con interpolación.

No obstante, cuando se trata de propiedades del DOM, podemos usar otro mecanismo llamado "Property binding". Se realiza poniendo el atributo entre corchetes, y dandole el valor de la expresión javascript, en nuestro caso la propiedad del componente.

¿Cuando usar uno u otro método? Ambos son válidos, pero por claridad, se suele usar la interpolación para el texto que se muestra, y el binding de propiedades para los atributos HTML. La cuestión es usar la sintaxis mas clara en cada caso.



El binding de propiedades es un binding de un solo sentido. ¿Qué quiere decir esto? Que la vista responde a los cambios del modelo, pero el modelo no se modifica si el control visual modifica el dato.

Binding de clases CSS

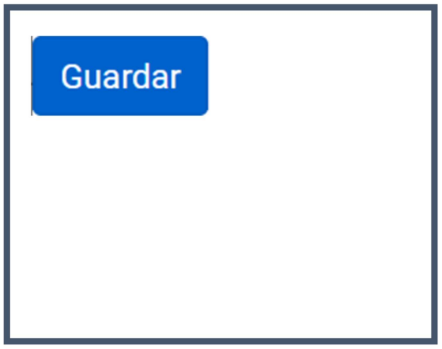
HTML

```
<button class="btn btn-
primary"
[class.active]="isActive">Gu
ardar</button>
```

TS

```
isActive = true;
```

Salida



El binding de clases permite añadir o eliminar clases CSS en base a campos del componente.

La expresión a usar debe evaluarse como cierta o falsa. Si es cierta, la clase CSS que sigue a “class.” se aplica.

Binding de estilos CSS

HTML

```
<h2
[style.backgroundColor]="isA
ctive? 'blue': 'gray'">Lista
de usuarios</h2>
```

TS

```
isActive = true;
```

Salida

El binding de estilos es similar al de clases CSS, pero se aplica a atributos CSS.

La expresión que se pasa se evalua y se asigna al atributo CSS especificado.

Directivas

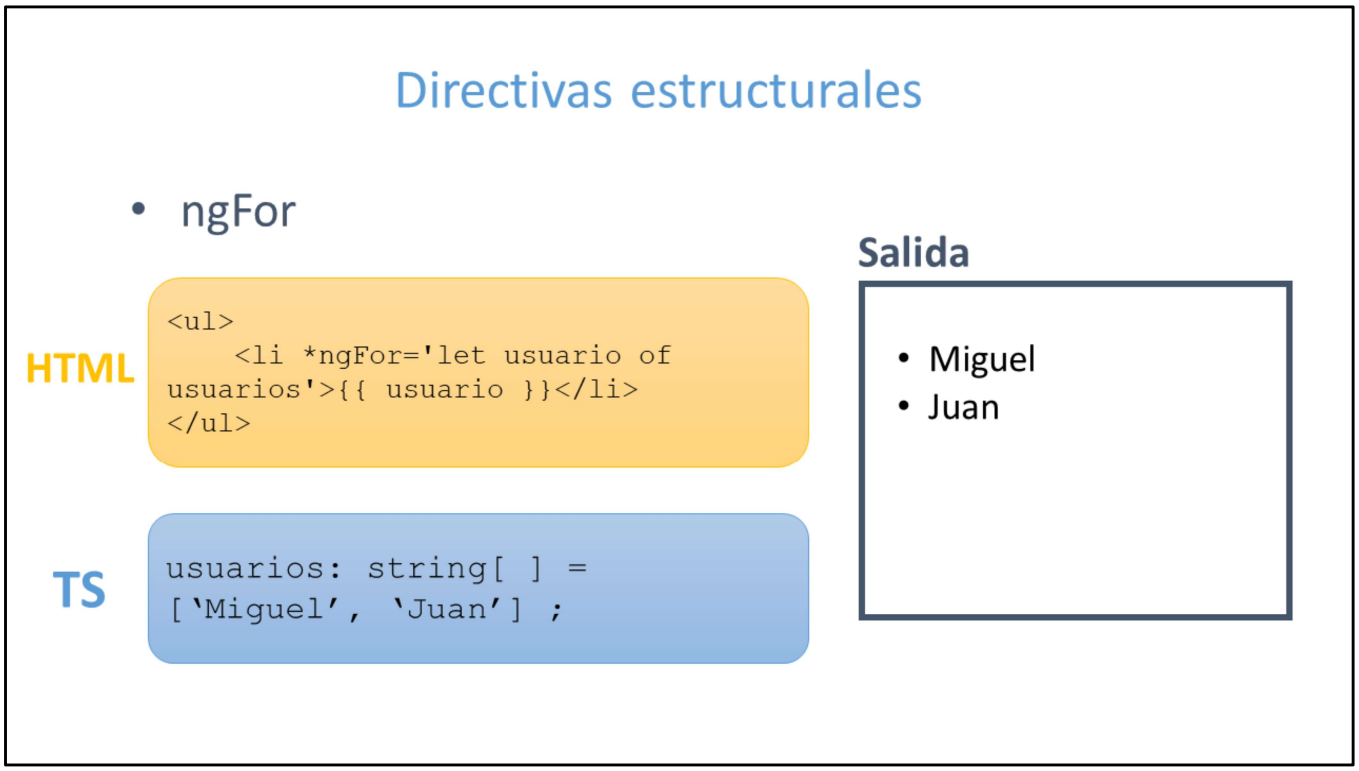
- Son instrucciones o comandos de Angular dentro de la plantilla HTML.
- Normalmente aparecen como atributos de un elemento

```
<li *ngFor="…"> </li>
```

- 2 tipos:
  - **Estructurales**: Modifican el DOM, crean o quitan nodos. Prefijo *
    - *ngFor*
    - *ngIf*
    - *ngSwitchCase*
  - **Atributo**: Alteran apariencia pero no modifican DOM

¿Qué son las directivas?

- Las directivas son instrucciones que usa Angular en las plantillas HTML, y que se representan como atributos de las etiquetas.
- Aunque parecen atributos HTML, no lo son. Normalmente empiezan por "ng"
- Hay dos tipos de directivas, las estructurales y las atributo, y normalmente las vemos en forma de etiquetas de elementos HTML como atributos.
  - Las directivas estructurales, que se diferencian fácilmente al ser precedidas por un asterisco, alteran el layout añadiendo, eliminando o reemplazando elementos en el DOM.
  - Las directivas atributo alteran la apariencia o el comportamiento de un elemento existente en el DOM y su aspecto es igual a la de atributos HTML



Esta directiva emula un bucle for para una colección de valores.

Angular repite el elemento tantas veces como valores tiene la colección indicada en la expresión.

En el ejemplo, la colección es "usuarios", y la variable que se pasa a cada iteración del bucle es 'usuario'.

Directivas estructurales

- ngFor (parámetros adicionales)

```
<ul>
    <li *ngFor='let usuario of usuarios; index as i'>{{ i }} - {{ usuario }}</li>
</ul>
```

Salida

- 0 - Miguel
- 1 - Juan

- index: indice dentro de la colección, empezando por 0
- first: cierto si es el primer elemento de la coleccion
- last: cierto si es el último elemento de la colección
- even: cierto si el indice del elemento es par
- odd: cierto si el indice del elemento es impar

https://angular.io/api/common/NgForOf

- Además de la variable "usuario", podemos pasar otras variables utiles (https://angular.io/api/common/NgForOf)
    - index: indice dentro de la colección, empezando por 0
    - first: cierto si es el primer elemento de la coleccion
    - last: cierto si es el último elemento de la colección
    - even: cierto si el indice del elemento es par
    - odd: cierto si el indice del elemento es impar

Directivas estructurales

- ngIf

HTML

```
<div *ngIf="usuarios.length == 0">
      No existen usuarios cargados en el sistema
</div>
```

TS

```
usuarios: string[ ] = ['Miguel', 'Juan'] ;
```

Habitualmente, necesitamos mostrar u ocultar un elemento HTML, en función de determinadas condiciones. Por ejemplo, si no existe ningun usuario en la lista, queremos ocultar la lista, e incluso mostrar un mensaje informando de que no existen usuarios en el sistema.

Para ello usamos la directiva "ngIf", que evalua una expresión booleana, y muestra u oculta el elemento que la contiene en base al resultado

Usamos el asterisco delante de ngIf, porque la directiva modifica la estructura del DOM.

ngIf no se limita a mostrar u ocultar el contenido con propiedades CSS de visibilidad, sino que directamente no se renderiza el contenido en el DOM, lo que es más efectivo y seguro.

Directivas estructurales

- ngIf (con bloques *then* y *else*)

"<condicion>; then *bloqueCierto* else *bloqueFalso*"

```
<div *ngIf="usuarios.length > 0; then listaUsuarios else noHayUsuarios"></div>

<ng-template #listaUsuarios>
    <ul>
        <li *ngFor='let usuario of usuarios'>{{ usuario }}</li>
    </ul>
</ng-template>

<ng-template #noHayUsuarios>
    <div>No existen usuarios cargados en el sistema</div>
</ng-template>
```

La sintaxis completa de la directiva ngIf es: "<condicion>; then *bloqueCierto* else *bloqueFalso*", donde bloqueCierto y bloqueFalso son variables de template asociadas a los bloques que se van a mostrar en cada caso.

Un bloque se define en la plantilla HTML con la etiqueta <ng-template>

Las variables template, permiten asociar el elemento HTML a una variable, que puede ser referenciada en cualquier parte de la plantilla HTML del componente.

Directivas estructurales

- ngSwitchCase

HTML

```
<div [ngSwitch] = "tabActiva">
    <div *ngSwitchCase="'lista'">
        Lista de usuarios…
    </div>
    <div *ngSwitchCase="'detalle'" >
        Detalle del usuario...
    </div>
    <div *ngSwitchDefault>
        Ninguna pestaña está activa
    </div>
</div>
```

TS

```
tabActiva = 'lista';
```

Esta directiva, actua de forma similar a las sentencias switch / case de los lenguajes de programación. Elije un camino u otro en función de un valor de una variable o expresión. Es más potente que ngIf, porque no se limita a la posibilidad cierto / falso.

Podemos usar esta directiva, por ejemplo, para implementar pestañas. Cada pestaña muestra un bloque de contenido, pero solo una es visible a la vez.

Fijarse que se realiza binding de la propiedad “ngSwitch” del div a una propiedad del componente, por eso va entre corchetes.

Binding de eventos

HTML

```
<input type="button" value="Aceptar" (click)="aceptar()">
```

TS

```
aceptar(){
   console.log("Boton pulsado");
}
```

https://www.w3schools.com/jsref/dom_obj_event.asp

Angular permite responder a eventos HTML, de una forma muy parecida a como lo hace el propio Javascript, es decir, especificando manejadores de eventos.

Para indicar un manejador de eventos, debemos usar el denominado "binding de eventos".

Entre paréntesis se indica el nombre del evento DOM HTML, según se puede ver en esta página: https://www.w3schools.com/jsref/dom_obj_event.asp)

Binding de eventos

- Acceso al evento HTML en el código TS:

HTML

```
<h1 (mousedown)="aceptar($event)">Pulse sobre el texto</h1>
```

TS

```
aceptar(evento){
   alert("Boton del raton pulsado: " + evento.which);
}
```

Si se desea acceder al evento dentro del método, agregar el parámetro $event al método

Binding de eventos

- Filtrado de eventos:

HTML

```
<input type="text" (keyup)="teclaPulsada()">

<input type="text" (keyup.enter)="teclaEnterPulsada()">
```

https://www.w3schools.com/jsref/event_key_key.asp

Para determinados eventos, podemos filtrar los eventos que manejamos en base al valor de alguno de los atributos del evento.

Por ejemplo, para los eventos de teclado (keyup, keydown...), podemos filtrar por los valores de la propiedad "key": "a", "enter", etc (https://www.w3schools.com/jsref/event_key_key.asp) .

De esta forma, solo se llamará al metodo "teclaEnterPulsada", cuando se pulse la tecla "Enter"

Binding de eventos

- Pasar valor del propio elemento HTML al manejador:

HTML

```
<input #nombre (keyup.enter)="enterPulsado(nombre.value)"
```

TS

```
enterPulsado(valor){
   alert("Valor del cuadro de texto: " + valor);
}
```

#nombre → Variable de plantilla (Template variable), que se asocia al elemento "input" HTML

- En los manejadores de eventos, es habitual necesitar acceder a propiedades del elemento HTML que ha lanzado el evento.
- Por ejemplo, al pulsar enter sobre un campo de texto, queremos mostrar en consola el valor del texto introducido.
- Podemos hacer esto de 2 formas:
  - Si usamos $event para pasar el evento al manejador, podemos usar la propiedad $event.target para acceder al elemento HTML, y por tanto a sus propiedades.
  - En lugar de eso, podemos usar las variables template, que permiten asociar el elemento HTML a una variable, que podemos pasar al manejador: