Templates angular – NGModel[()]

Variables de plantilla

Las variables de plantilla le ayudan a utilizar datos de una parte de una plantilla en otra parte de la plantilla. Utilice variables de plantilla para realizar tareas como responder a la entrada del usuario o ajustar con precisión los formularios de su aplicación.

Una variable de plantilla puede hacer referencia a lo siguiente:

* un elemento DOM dentro de una plantilla
* una directiva
* un elemento
* [TemplateRef](https://angular.io/api/core/TemplateRef)
* a [componente web](https://developer.mozilla.org/en-US/docs/Web/Web_Components)

Sintaxis

En la plantilla, utiliza el símbolo de almohadilla #, para declarar una variable de plantilla. La siguiente variable de plantilla #phone, declara una phonevariable en un <input>elemento.

src / app / app.component.html

<input #phone placeholder="phone number" />

Consulte una variable de plantilla en cualquier parte de la plantilla del componente. Aquí, un poco <button>más abajo en la plantilla se refiere a la phonevariable.

src / app / app.component.html

<input #phone placeholder="phone number" />

<!-- lots of other elements -->

<!-- phone refers to the input element; pass its `value` to an event handler -->

<button (click)="callPhone(phone.value)">Call</button>

Cómo Angular asigna valores a las variables de la plantilla

Angular asigna a una variable de plantilla un valor basado en dónde declaras la variable:

* Si declara la variable en un componente, la variable se refiere a la instancia del componente.
* Si declara la variable en una etiqueta HTML estándar, la variable se refiere al elemento.
* Si declara la variable en un [<ng-template>](https://angular.io/api/core/ng-template)elemento, la variable se refiere a una [TemplateRef](https://angular.io/api/core/TemplateRef)

instancia, que representa la plantilla. Para obtener más información sobre [<ng-template>](https://angular.io/api/core/ng-template), consulte [Cómo usa Angular la \*sintaxis del asterisco ,,](https://angular.io/guide/structural-directives#asterisk) en las [directivas estructurales](https://angular.io/guide/structural-directives) .

* Si la variable especifica un nombre en el lado derecho, como , la variable se refiere a la directiva o componente en el elemento con un nombre coincidente . #var="[ngModel](https://angular.io/api/forms/NgModel)"exportAs

Usar [NgForm](https://angular.io/api/forms/NgForm) con variables de plantilla

En la mayoría de los casos, Angular establece el valor de la variable de plantilla en el elemento en el que ocurre. En el ejemplo anterior, se phonerefiere al número de teléfono <input>. El controlador de clic del botón pasa el <input>valor al callPhone()método del componente .

La [NgForm](https://angular.io/api/forms/NgForm)directiva demuestra cómo obtener una referencia a un valor diferente haciendo referencia al exportAsnombre de una directiva . En el siguiente ejemplo, la variable de plantilla itemForm, aparece tres veces separada por HTML.

src / app / hero-form.component.html

<form #itemForm="[ngForm](https://angular.io/api/forms/NgForm)" (ngSubmit)="onSubmit(itemForm)">

<label for="name">Name</label>

<input type="text" id="name" class="form-control" name="name" [ngModel](https://angular.io/api/forms/NgModel) required />

<button type="submit">Submit</button>

</form>

<div [hidden]="!itemForm.form.valid">

<p>{{ submitMessage }}</p>

</div>

Sin el [ngForm](https://angular.io/api/forms/NgForm) valor del atributo, el valor de referencia de itemForm sería el[HTMLFormElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLFormElement), <form>. Sin embargo, existe una diferencia entre a [Component](https://angular.io/api/core/Component)y

a [Directive](https://angular.io/api/core/Directive)en que Angular hace referencia a a [Component](https://angular.io/api/core/Component)sin especificar el valor del atributo, y a [Directive](https://angular.io/api/core/Directive)no cambia la referencia implícita o el elemento.

Con [NgForm](https://angular.io/api/forms/NgForm), itemFormes una referencia a la directiva [NgForm](https://angular.io/api/forms/NgForm" \o "API: NgForm) con la capacidad de rastrear el valor y la validez de cada control en el formulario.

A diferencia del <form>elemento nativo , la [NgForm](https://angular.io/api/forms/NgForm)directiva tiene una formpropiedad. La [NgForm](https://angular.io/api/forms/NgForm) formpropiedad le permite deshabilitar el botón enviar si itemForm.form.validno es válido.

Alcance de la variable de plantilla

Consulte una variable de plantilla en cualquier lugar dentro de su plantilla circundante. [Las directivas estructurales](https://angular.io/guide/built-in-directives) , como y , o actúan como un límite de plantilla. No puede acceder a las variables de plantilla fuera de estos límites.\*[ngIf](https://angular.io/api/common/NgIf)\*[ngFor](https://angular.io/api/common/NgForOf)[<ng-template>](https://angular.io/api/core/ng-template)

Defina una variable solo una vez en la plantilla para que el valor de tiempo de ejecución siga siendo predecible.

Accediendo en una plantilla anidada

Una plantilla interna puede acceder a las variables de plantilla que define la plantilla externa.

En el siguiente ejemplo, cambiando el texto en el <input>cambia el valor en el <span>porque angular actualiza inmediatamente cambios a través de la variable de plantilla, ref1.

src / app / app.component.html

content\_copy<input #ref1 type="text" [([ngModel](https://angular.io/api/forms/NgModel))]="firstExample" />

<span \*[ngIf](https://angular.io/api/common/NgIf)="true">Value: {{ ref1.value }}</span>

En este caso, hay un implícito [<ng-template>](https://angular.io/api/core/ng-template)alrededor de <span>y la definición de la variable está fuera de él. El acceso a una variable de plantilla desde la plantilla principal funciona porque la plantilla secundaria hereda el contexto de la plantilla principal.

Reescribir el código anterior en una forma más detallada muestra explícitamente el [<ng-template>](https://angular.io/api/core/ng-template).

content\_copy<input #ref1 type="text" [([ngModel](https://angular.io/api/forms/NgModel))]="firstExample" />

<!-- New template -->

<ng-template [[ngIf](https://angular.io/api/common/NgIf)]="true">

<!-- Because the context is inherited, the value is available to the new template -->

<span>Value: {{ ref1.value }}</span>

</ng-template>

Sin embargo, acceder a una variable de plantilla desde fuera de la plantilla principal no funciona.

content\_copy<input \*[ngIf](https://angular.io/api/common/NgIf)="true" #ref2 type="text" [([ngModel](https://angular.io/api/forms/NgModel))]="secondExample" />

<span>Value: {{ ref2?.value }}</span> <!-- doesn't work -->

La forma detallada muestra que ref2está fuera de la plantilla principal.

content\_copy<ng-template [[ngIf](https://angular.io/api/common/NgIf)]="true">

<!-- The reference is defined within a template -->

<input #ref2 type="text" [([ngModel](https://angular.io/api/forms/NgModel))]="secondExample" />

</ng-template>

<!-- ref2 accessed from outside that template doesn't work -->

<span>Value: {{ ref2?.value }}</span>

Considere el siguiente ejemplo que usa .\*[ngFor](https://angular.io/api/common/NgForOf)

<ng-container \*[ngFor](https://angular.io/api/common/NgForOf)="let i of [1,2]">

<input #ref type="text" [value]="i" />

</ng-container>

{{ ref.value }}

Aquí, ref.valueno funciona. La directiva estructural crea una instancia de la plantilla dos veces porque itera sobre los dos elementos de la matriz. Es imposible definir qué significa la referencia.\*[ngFor](https://angular.io/api/common/NgForOf)\*[ngFor](https://angular.io/api/common/NgForOf)ref.value

Con directivas estructurales, como o , no hay forma de que Angular sepa si alguna vez se crea una instancia de una plantilla.\*[ngFor](https://angular.io/api/common/NgForOf)\*[ngIf](https://angular.io/api/common/NgIf)

Como resultado, Angular no puede acceder al valor y devuelve un error.

Accediendo a una variable de plantilla dentro [<ng-template>](https://angular.io/api/core/ng-template)

Cuando declaras la variable en un [<ng-template>](https://angular.io/api/core/ng-template), la variable se refiere a una [TemplateRef](https://angular.io/api/core/TemplateRef)instancia, que representa la plantilla.

src / app / app.component.html

content\_copy<ng-template #ref3></ng-template>

<button (click)="log(ref3)">Log type of #ref</button>

En este ejemplo, al hacer clic en el botón se llama a la log()función, que envía el valor de #ref3a la consola. Debido a que la #refvariable está en an [<ng-template>](https://angular.io/api/core/ng-template), el valor es [TemplateRef](https://angular.io/api/core/TemplateRef).

La siguiente es la salida expandida de la consola del navegador de la [TemplateRef](https://angular.io/api/core/TemplateRef)()función con el nombre de [TemplateRef](https://angular.io/api/core/TemplateRef).

content\_copy▼ ƒ TemplateRef()

name: "TemplateRef"

\_\_proto\_\_: Function

Variable de entrada de plantilla

Una *variable de entrada de plantilla* es una variable a la que se hace referencia dentro de una única instancia de la plantilla. Declara una variable de entrada de plantilla utilizando la letpalabra clave como en let hero.

Hay varias de estas variables en este ejemplo: hero, iy odd.

content\_copy<ng-template #hero let-hero let-i="index" let-odd="isOdd">

<div [class]="{'odd-row': odd}">{{i}}:{{hero.name}}</div>

</ng-template>

El alcance de la variable está limitado a una sola instancia de la plantilla repetida. Use el mismo nombre de variable nuevamente en la definición de otras directivas estructurales.

Por el contrario, declara una variable de plantilla anteponiendo el nombre de la variable con #, como en #var. Una variable de plantilla se refiere a su elemento, componente o directiva adjunta.

Las variables de entrada de plantilla y los nombres de las variables de plantilla tienen sus propios espacios de nombres. La variable de entrada de plantilla heroen let heroes distinta de la variable de plantilla heroen #hero