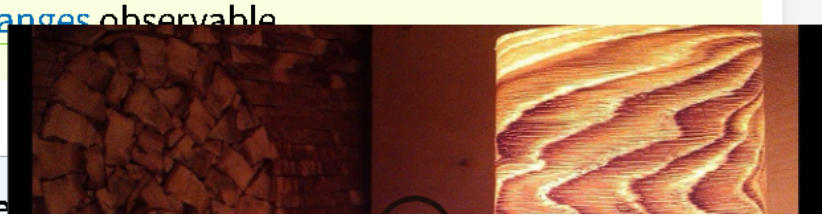
NgModelChange & Change Event In Angular

4 Comments / March 9, 2023 / 3 minutes of reading

⟵ Two way binding Angular Tutorial Child/Nested Components ⟶

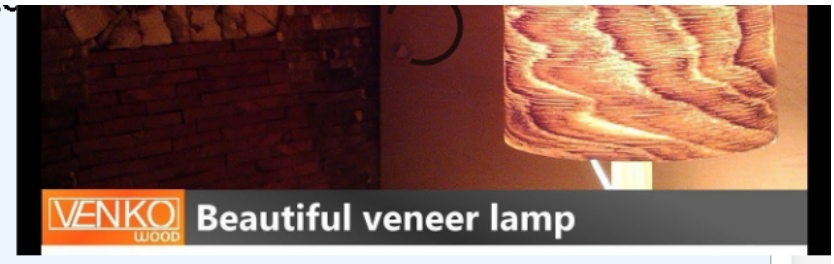
NgModelChange is an Angular specific event, which we can use to listen for changes to the user input. It is the @Output property of the ngModel directive, Hence we need to use it along with it. `ngModle` raises the `NgModelChange` event, whenever the model changes. Another way to listen for change is to use the `change` DOM event. We also learn how to use them and also the difference between `change` & `ngModelChange`.

If you are using template driven forms or reactive forms, then the better way to listen for changes is using the ValueChanges observable.



Table

NgModelChange Example
ngModel
ngModelChange with ngModel
Change Event
Change Event Example
NgModelChange Vs Change
References



NgModelChange Example

The following is the simple example of `ngModelChange`.

We assign the method in the component class (handler function) to the `ngModelChange` using the event binding syntax

```
//Template

Name:
<input type="text" name="name" ngModel (ngModelChange)="nameChanged($event)">

```

`nameChanged` is the handler function, which we need to define in the component class. We can access the *new value* by using the `$event` as an argument to the handler function.

```
//Component
nameChanged(arg) {
  console.log("modelchanged " + arg);
}

```

ngModel

We usually use the ngModel as follows to achieve the two-way binding. `[(ngModel)]="email"` keeps the email property in the component class in sync with the template.

```
<input type="text" name="email" [(ngModel)]="email">
```

Internally, Angular converts the above syntax to the following syntax.

```
<input [ngModel]="email" (ngModelChange)="email = $event">
```

The component property `email` is bound to the `input` element using the property binding (`[ngModel]="email"`). Any changes made to the input is updated in the component using the `(ngModelChange)="email = $event"` event binding.

ngModelChange with ngModel

Consider the following example.

```
<input [(ngModel)]="firstName" (ngModelChange)="firstNameChanged($event)"/>
```

The Angular converts the above to the following syntax. We end up with the two `ngModelChange` event bindings.

```
<input [ngModel]="firstName (ngModelChange)="firstName = $event"   (ngModelChange)="fir
```

Here the `ngModelChange` fires in the order it is specified. Hence the

`(ngModelChange)="firstName = $event"` fires first.

`(ngModelChange)="firstNameChanged($event)"` fires later.

Hence in the component class, the arg & this.firstName is always the same.

```

firstName
;

firstNameChanged(arg) {
  console.log(
      "firstNameChanged  argument " + arg + "  component " + this.firstName
  );
}

```

But if you put `ngModelChange` ahead of the `ngModel` as in the example below

```

<input (ngModelChange)="lastNameChanged($event)" [(ngModel)]="lastName" />


```

Angular internally converts it as follows

```

<input (ngModelChange)="lastNameChanged($event)" [ngModel]="lastName" (ngModelChange

```

Here `(ngModelChange)="lastNameChanged($event)"` fires first. Hence in the component class `arg` contains the latest value of the, while `this.lastName` **still holds the previous value**

```

lastName
;

lastNameChanged(arg) {
```

```
  console.log(
     "lastNameChanged  argument " + arg + "  component " + this.lastName
  );
}

```

Change Event

The `(change)` is a DOM event fires when changes to the form fields like `<input>`, `<select>`, and `<textarea>` is committed by the user.

This event fires when

- user changes the input & moves the focus away from the text box (blur event)
- On `<select>` it fires when the user selects a new option either by a mouse click or using a keyboard.
- Fires when the state of a check box or radio button change due to users action

Change Event Example

The following example shows how to use the `change` event in Angular.

```

Name
<input type="text" name="name1" (change)="name1Changed($event)">


<br>
country
<select name="country1" (change)="country1Changed($event)" >
  <option [ngValue]="null" disabled>Select Country</option>
  <option *ngFor="let country of countries" [ngValue]="country.id">{{country.name}}</optior
</select>

```

Component class

```
name1Changed(arg) {
 console.log("name1Changed " + arg.target.value);
 console.log(arg);
}

country1Changed(arg) {
  console.log("country1Changed " + arg.target.value);
  console.log(arg);
}

```

The change event for text element fires when we move the focus away from the element (blurred the input). This is different from the ngModelChange, which fires the event for each input change.

The other import point is the `$event` parameter. In the `ngModelChange`, it is always the new value. But in the case of a change event, it is event data. The event data is an object containing data about the event. We need to use the `target.value` to access the value.

NgModelChange Vs Change

NgModelChange	Change
NgModelChange is Angular specific event	Change is a DOM Event and has nothing to do with the Angular.
We must use the ngModelChange along with the ngModel directive	You can use change event on <input>, <select>, and <textarea> form elements.
ngModelChange event passes new value	Change event passes event parameter, Use the `target.value` to access the new value
ngModelChange will trigger with each input change	Change event fires when you remove the focus from input text after changing the content.

References

ngModel

ngModel.

Source Code

Read More

1. Angular Tutorial
2. Typescript Tutorial
3. ngModel & Two-way binding
4. ValueChanges
5. Template Driven Forms
6. Reactive Forms

← Two way binding | Angular Tutorial | Child/Nested Components →

4 thoughts on "NgModelChange & Change Event in Angular"

KARTHICK

DECEMBER 4, 2022 AT 3:54 PM

Hi, Under "ngModelChange with ngModel"

(I think 'closing double quote' need to add)

Reply

RAHUL VERMA

JUNE 24, 2022 AT 4:33 PM

how to dynamic append and remove html on the change of multiple select
how to do in angular 13?

Reply

ANONYMOUS
JUNE 24, 2022 AT 4:37 PM

urgent, tell soon

Reply

JULIAN HONMA
NOVEMBER 11, 2020 AT 9:04 PM

If nothing seems to work, check the terminal window where you launched the server with `ng serve`. If you see this error `Can't bind to 'ngModel' since it isn't a known property of 'input'.`, then check `app.module.ts` in `@NgModule > imports` to see if `FormsModule` is present.

(Other subject) The point the author is making at the end of the section **ngModelChange with ngModel** is that you should not put the `(ngModelChange)` before `[(ngModel)]`. Here are my logs showing the correct order for firstName, and incorrect order for lastName:

```
app.component.ts:23 firstNameChanged {arg: "F", this.firstName: "F"}
app.component.ts:23 firstNameChanged {arg: "Fi", this.firstName: "Fi"}
app.component.ts:23 firstNameChanged {arg: "Fir", this.firstName: "Fir"}
```

```
app.component.ts:23 firstNameChanged {arg: "Firs", this.firstName: "Firs"}
app.component.ts:23 firstNameChanged {arg: "First", this.firstName: "First"}
app.component.ts:27 lastNameChanged {arg: "L", this.lastName: undefined}
app.component.ts:27 lastNameChanged {arg: "La", this.lastName: "L"}
app.component.ts:27 lastNameChanged {arg: "Las", this.lastName: "La"}
```

And I just noticed, but if you have any problem, it looks like the author adds a link to his code in **Reference** > Source code at the end of the page before the Comment section, and you can see his code and preview the result.

Reply

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Post Comment »

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Copyright © 20