# How to use [(ngModel)] on div's contenteditable in angular2?

39

16

I am trying to use ngModel to two way bind div's contenteditable input content as follows:

```
<div id="replyiput" class="btn-input"  [(ngModel)]="replyContent"
contenteditable="true" data-text="type..." style="outline: none;"    ></div>
```

but it is not working and an error occurs:

```
EXCEPTION: No value accessor for '' in [ddd in PostContent@64:141]
app.bundle.js:33898 ORIGINAL EXCEPTION: No value accessor for ''
```

angular ionic2 contenteditable

edited Aug 22 '17 at 4:10
danday74 21.5k 17 108 143

asked Feb 13 '16 at 9:11
Kim Wong 633 3 11 18

## 6 Answers

75

`NgModel` expects the bound element to have a `value` property, which `div` s don't have. That's why you get the `No value accessor` error.

You can set up your own equivalent property and event databinding using the `textContent` property (instead of `value` ) and the `input` event:

```
import {Component} from 'angular2/core';
@Component({
  selector: 'my-app',
  title = 'Angular 2 RC.4';
  model = 'some text';
  constructor() { console.clear(); }
}
```

Plunker

I don't know if the `input` event is supported on all browsers for `contenteditable`. You could always bind to some keyboard event instead.

edited Jul 5 '16 at 15:00

answered Feb 13 '16 at 18:19

Mark Rajcok
**303k** 95 444 465

Thank you for your answer. But it is not a two way binding. When the user type something in the input, the "model" var will not change. – Kim Wong Feb 14 '16 at 12:23

1 @KimWong, the `model` var is definitely changing in the Plunker I provided. That's why I put `{{model}}` in the view/template, so that we can see it change when we edit the div. – Mark Rajcok Feb 15 '16 at 15:47

4 Regardless of the event used to trigger model=$event.target.textContent, this currently doesn't work properly on Firefox and Edge. The cursor is always set at index 0 when typing. You should be aware of this. – Lys Apr 3 '17 at 14:00

4 guys, anyone know how to sort out so the cursor index to not be set at 0 all the time? – Chris Tarasovs Jun 4 '17 at 17:35

1 currently this is only useful for typing backwards – rrrafalsz Mar 5 '18 at 19:56

12

**Updated answer (2017-10-09)**:

Now I have ng-contenteditable module. Its compatibility with Angular forms.

**Old answer (2017-05-11)**: In my case, I can simple to do:

```
<div
  contenteditable="true"
```

First time, after `ngOnInit()` and get `post` from backend, I set `this.postTitle = post.postTitle` in my component.

edited Oct 9 '17 at 18:10

answered May 11 '17 at 10:09

ktretyak
4,687 5 22 41

---

9

Working Plunkr here http://plnkr.co/edit/j9fDFc, but relevant code below.

Binding to and manually updating `textContent` wasn't working for me, it doesn't handle line breaks (in Chrome, typing after a line break jumps cursor back to the beginning) but I was able to get it work using a contenteditable model directive from https://www.namekdev.net/2016/01/two-way-binding-to-contenteditable-element-in-angular-2/.

I tweaked it to handle multi-line plain text (with `\n` s, not `<br>` s) by using `white-space: pre-wrap` , and updated it to use `keyup` instead of `blur` . Note that some solutions to this problem use the `input` event which isn't supported on IE or Edge on `contenteditable` elements yet.

Here's the code:

**Directive:**

```
import {Directive, ElementRef, Input, Output, EventEmitter, SimpleChanges} from
'angular2/core';

@Directive({
  selector: '[contenteditableModel]',
  host: {
    '(keyup)': 'onKeyup()'
  }
})
export class ContenteditableModel {
  @Input('contenteditableModel') model: string;
  @Output('contenteditableModelChange') update = new EventEmitter();

  /**
  private lastViewModel: string;

  constructor(private elRef: ElementRef) {
  }

  ngOnChanges(changes: SimpleChanges) {
    if (changes['model'] && changes['model'].currentValue !== this.lastViewModel) {
      this.lastViewModel = this.model;
      this.refreshView();
    }
  }

  /** This should probably be debounced. */
  onKeyup() {
    var value = this.elRef.nativeElement.innerText;
    this.lastViewModel = value;
    this.update.emit(value);
  }

  private refreshView() {
    this.elRef.nativeElement.innerText = this.model
  }
}
```

**Usage:**

```
import {Component} from 'angular2/core'
import {ContenteditableModel} from './contenteditable-model'

@Component({
  selector: 'my-app',
  providers: [],
  directives: [ContenteditableModel],
  styles: [
    `div {
      white-space: pre-wrap;

      /* just for looks: */
      border: 1px solid coral;
      width: 200px;
      min-height: 100px;
    <b>Output:</b>
    <div>{{text}}</div>

    <b>Input:</b><br>
    <button (click)="text='Success!'">Set model to "Success!"</button>
  `
})
export class App {
  text: string;

  constructor() {
    this.text = "This works\nwith multiple\n\nlines"
  }
}
```

Only tested in Chrome and FF on Linux so far.

edited Dec 21 '16 at 3:05

answered Dec 21 '16 at 1:45
tobek
2,664 2 19 34

1 Tested on Firefox with Windows as well, under Ionic 2, and your code works there as well. Thanks! – Cel Apr 20 '17 at 8:55

7

Here's another version, based on @tobek's answer, which also supports html and pasting:

```
import {
  Directive, ElementRef, Input, Output, EventEmitter, SimpleChanges, OnChanges,
  HostListener, Sanitizer, SecurityContext
} from '@angular/core';

@Directive({
  selector: '[contenteditableModel]'
})
export class ContenteditableDirective implements OnChanges {
  /** Model */
  private elRef: ElementRef,
  private sanitizer: Sanitizer
 ) { }

 ngOnChanges(changes: SimpleChanges) {
  if (changes['contenteditableModel']) {
   // On init: if contenteditableModel is empty, read from DOM in case the element
has content
   if (changes['contenteditableModel'].isFirstChange() && !this.contenteditableModel)
{
    this.onInput(true);
   }
   this.refreshView();
  }
 }

 @HostListener('input') // input event would be sufficient, but isn't supported by IE
 @HostListener('blur')  // additional fallback
 @HostListener('keyup') onInput(trim = false) {
  let value = this.elRef.nativeElement[this.getProperty()];
  if (trim) {
   value = value.replace(/^[\n\s]+/, '');
   value = value.replace(/[\n\s]+$/, '');
  }
  this.contenteditableModelChange.emit(value);
 }

 @HostListener('paste') onPaste() {
  this.onInput();
  if (!this.contenteditableHtml) {
   // For text-only contenteditable, remove pasted HTML.
   // 1 tick wait is required for DOM update
   setTimeout(() => {
    if (this.elRef.nativeElement.innerHTML !== this.elRef.nativeElement.innerText) {
     this.elRef.nativeElement.innerHTML = this.elRef.nativeElement.innerText;
    }
   });
  }
 }

 private refreshView() {
  const newContent = this.sanitize(this.contenteditableModel);
  // Only refresh if content changed to avoid cursor loss
  private getProperty(): string {
    return this.contenteditableHtml ? 'innerHTML' : 'innerText';
  }

  private sanitize(content: string): string {
    return this.contenteditableHtml ? this.sanitizer.sanitize(SecurityContext.HTML,
content) : content;
  }
}
```

answered Jun 28 '17 at 12:45
Rene Hamburger
**1,186** 8 8

Thanks, but to avoid **ExpressionChangedAfterItHasBeenCheckedError** please use asynchronous `EventEmitter` in output `@Output() contenteditableModelChange?= new EventEmitter(true);` reference to article. Maybe you can update your code. – Atiris Sep 22 '17 at 7:56

This should be the accepted answer. Excellent. – Charles Robertson Apr 21 at 12:14

4

I've fiddled around with this solutions and will use the following solution in my project now:

```
<div #topicTitle contenteditable="true" [textContent]="model"
(input)="model=topicTitle.innerText"></div>
```

I prefer using the template reference variable to the "$event" stuff.

Related link: https://angular.io/guide/user-input#get-user-input-from-a-template-reference-variable

answered Nov 6 '17 at 21:03
Flo
**53** 6

I used this solution on an editable TD as well. {{model}} as suggested by some other solutions gave me issues while typing. It would dynamically update

1

Here is a simple solution if what you are binding to is a string, no events necessary. Just put a text box input inside the table cell and bind to that. Then format your text box to transparent

HTML:

```
<tr *ngFor="let x of tableList">
    <td>
        <input type="text" [(ngModel)]="x.value" [ngModelOptions]="{standalone: true}">
    </td>
</tr>
```

answered Jun 8 '18 at 19:19
Isaac
**23** 3