

# What is the difference between parentheses, brackets and asterisks in Angular2?

Asked 3 years, 4 months ago   Active 1 year ago   Viewed 55k times



105



40

I have been reading the Angular 1 to 2 quick reference in the [Angular website](#), and one thing I didn't completely understand was the difference between these special characters. For example one that uses asterisks:

```
<tr *ngFor="#movie of movies">
  <td>{{movie.title}}</td>
</tr>
```

I understand here that the hash (#) symbol defines `movie` as a local template variable, but what does the asterisk before `ngFor` mean? And, is it necessary?

Next, are the examples that use brackets:

```
<a [routerLink]="['Movies']">Movies</a>
```

I somewhat understand that the brackets around `routerLink` bind it to that HTML attribute / Angular directive. Does this mean that they are a pointer for Angular to evaluate an expression? Like `[id]="movieId"` would be the equivalent of `id="movie-{{movieId}}"` in Angular 1?

Lastly, are parentheses:

```
<button (click)="toggleImage($event)">
```

Are these only used for DOM events and can we use other events like `(load)="someFn()"` or `(mouseenter)="someFn()"` ?

I guess the real question is, do these symbols have a special meaning in Angular 2, and what is the easiest way to know **when to use each one**? Thanks!!





## 4 Answers



103



All details can be found here: <https://angular.io/docs/ts/latest/guide/template-syntax.html>

- `directiveName` - is the short hand form for structural directives where the long form can only be applied to `<template>` tags. The short form implicitly wraps the element where it's applied in a `<template>`.
- `[prop]="value"` is for object binding to properties ( `@Input()` of an Angular component or directive or a property of a DOM element). There are special forms:
  - `[class.className]` binds to a css class to enable/disable it
  - `[style.stylePropertyName]` binds to a style property
  - `[style.stylePropertyName.px]` binds to a style property with a preset unit
  - `[attr.attrName]` binds a value to an attribute (visible in the DOM, while properties are not visible)
  - `[role.roleName]` binds to the ARIA role attribute (not yet available)
- `prop="{{value}}"` binds a value to a property. The value is stringified (aka interpolation)
- `(event)="expr"` binds an event handler to an `@Output()` or DOM event
- `#var` or `*var` has different functions depending on the context
  - ~~In an `*ngFor="#x in y; #i=index"` scope variables for the iteration are created~~ (In beta.17 this is changed to `*ngFor="let x in y; let i=index"`)
  - On a DOM element `<div #mydiv>` a reference to the element
  - On an Angular component a reference to the component
  - On an element that is an Angular component or has an Angular directive where `exportAs:"ngForm"` is defined, `#myVar="ngForm"` creates a reference to this component or directive.



edited May 1 '18 at 7:59

answered Mar 11 '16 at 16:14



Günter Zöchbauer

358k 82 1137 1037

- 12 Or bind- for [] and on- for () or <template [ngFor]> for \*ngFor . – [Günter Zöchbauer](#) Mar 13 '16 at 7:13 
- 4 What does [(ngModel)] mean i.e. parenthesis within square brackets? – [Undefined](#) Aug 28 '17 at 21:21
- 7 Two-way binding (also called "banana in a box6). It's the combination (or short form of) of [ngModel]="foo" (ngModelChange)="foo = \$event" where the first part updates the ngModel property ( @Input() ngModel; of the NgModel directive ) when foo` changes and the 2nd part updates foo when the @Output() ngModelChange; (of the NgModel directive) emits an event. NgModel is used to bind values to form elements and components. [(bar)] can be used for any @Input() bar; @Output() barChange; combination, also of your own components. – [Günter Zöchbauer](#) Aug 29 '17 at 3:08
- 2 @DiPix [prop]="value" can assign values of any type, prop="{{value}}" always stringifies value before assignment and is therefore useless to pass objects. – [Günter Zöchbauer](#) May 15 '18 at 10:18
- 1 @DiPix no, doesn't matter, because HTML can only render string anyway, but if you want to pass a value to an @Input() that is not of type string , then {{value}} won't work. Also some properties (in contrary to attributes) of native HTML elements are not of type string and can cause issues if set with {{{}}}. You're always safe with [prop]="value" , at least I don't know an example where this would cause issues and prop={{value}} would not. – [Günter Zöchbauer](#) May 15 '18 at 10:34 

**[] - Property binding** One-way from data source to view target. eg

22

```
{{expression}}
[target]="expression"
bind-target="expression"
```

[Run code snippet](#)
[Expand snippet](#)

We can use bind- instead of []

**() -> Event Binding** One-way from view target to data source

```
(target)="statement"
on-target="statement"
```

[Run code snippet](#)
[Expand snippet](#)

We can use on- instead of ()

## [()]- Two way Binding Banana in a box

```
[(target)]= "expression"  
bindon-target="expression"
```

[Run code snippet](#)[Expand snippet](#)

We can use bindon- instead of [()]

edited Jun 8 '18 at 9:59

answered Jun 8 '18 at 9:54



Shajin Chandran

396 3 6

As mentioned already, the Angular documentation, especially the "hero tutorial", explains this deeper. [Here is the link if you want to check it out.](#)

12

Parentheses are events of the element you are working on, like the click on a button like your example; this could also be mousedown, keyup, onselect or any action/event for that element, and what is after the = is the name of the method to call -- using the parenthesis for the call. That method should be defined on your component class, i.e.:

```
<element (event)="method()"></element>
```

Brackets works the other way. They are to get data from your class -- the opposite of the parenthesis that were sending the event -- so a common example is the usage of a style like this:

```
<element [ngStyle]="{display:someClassVariable}">
```

See? You are giving the element a style based on your model/class.

For this you could have used...

```
<element style="display:{{ModelVariable}};">
```

The recommendation is that you use double curly brackets for things that you will print on the screen like:

```
<h1>{{Title}}</h1>
```

Whatever you use, if you are consistent, it will help the readability of your code.

Lastly, for your \* question, it is a longer explanation, but it is very VERY important: It abstracts some methods' implementation that otherwise you would have to do to get an `ngFor` to work.

One **important update** is that in the `ngFor` you will no longer use `hash`; you need to use `let` instead as follows:

```
<tr *ngFor="let movie of movies">
  <td>{{movie.title}}</td>
</tr>
```

One last thing worth mentioning is that all of the above applies also for your components, e.g. if you create a method in your component, it will be called using `()` :

```
<my-owncomponent
  (onSearched)="MethodToCall()"
  [MyInputData]="SearchParamsArray"></my-owncomponent>
```

edited Jul 25 '18 at 15:26



ruffin

9,935 5 56 102

answered Aug 10 '16 at 6:02



Gary

364 4 7

Yes, they do have special meaning. The easiest way is to read docs.

9

Angular2 [docs have good explanation](#) for all this:

- star - <https://angular.io/guide/template-syntax#structural-directives>
- [] - <https://angular.io/guide/template-syntax#property-binding--property->
- () - <https://angular.io/guide/template-syntax#event-binding---event->
- hash - <https://angular.io/guide/template-syntax#interpolation---->

edited Mar 23 '18 at 17:38

kairos

answered Mar 11 '16 at 16:19



Alexander Trakhimenok



95 8



3,950 2 17 42

- 
- 6 Yes, reading the manual is good, but this isn't an answer. It's a collection of links from which the reader may figure out the answer... – [Jasper](#) Aug 29 '17 at 9:41
- 
- 1 My answer precisely answer the 2 questions asked: "I guess the real question is, do these symbols have a special meaning in Angular 2, and what is the easiest way to know when to use each one?" - 1) have special meaning. 2) Docs are the easiest way. – [Alexander Trakhimenok](#) Aug 29 '17 at 10:05
- 
- 3 Technically, that's correct. In practice, though, the real question is "what are the differences between each of those", and that's the question that should be answered. – [Jasper](#) Aug 29 '17 at 10:14
- 
- Everybody knows that. People come here to look for a cheatsheet. – [Qian Chen](#) Feb 28 at 8:25
-