

# Google Angular, API Doc

1. Content Structure
2. Interaction
3. Spacing Guide
4. UI Components and Typography Guide

\* All copy in this doc is for placeholder purpose. Exact copy is TBD.

# 1. Content Structure

SEARCH DOCS...

DOCS HOME

5 MIN QUICKSTART

TUTORIAL

DEVELOPER GUIDES

TESTING GUIDES

API PREVIEW

ANGULAR RESOURCES

HELP &amp; SUPPORT

# NgSwitch

Directive

Stability: Stable

Angular 2 for TypeScript ▾

## WHAT IT DOES

The `NgSwitch` directive is used to conditionally swap DOM structure on your template based on a scope expression. Elements within `NgSwitch` but without `NgSwitchWhen` or `NgSwitchDefault` directives will be preserved at the location as specified in the template.

## HOW TO USE

```
<!-- platform directive - no import needed -->
<div [ng-switch]="conditionExpression">
  <template [ng-switch-when]="case1Exp">...</template>
  <template ng-switch-when="case2LiteralString">...</template>
  <template ng-switch-default>...</template>
</div>
```

## SELECTORS ?

[ng-switch]

## OUTPUTS ?

ng-model-change bound to NgFormControl.update

## INPUTS ?

ng-switch bound to NgSwitch.ngSwitch

## EXPORTED AS ?

form

## CLASS

```
class NgSwitch {
  ngSwitch
}
```

## OVERVIEW

`NgSwitch` simply chooses nested elements and makes them visible based on which element matches the value obtained from the evaluated expression. In other words, you define a container element (where you place the directive), place an expression on the

[ng-switch] = "... " attribute), define any inner elements inside of the directive and place a [ng-switch-when] attribute per element. The when attribute is used to inform NgSwitch which element to display when the expression is evaluated. If a matching expression is not found via a when attribute then an element with the default attribute is displayed.

live demo ↗

1. <ANY [ng-switch] = "expression">
2. <template [ng-switch-when] = "whenExpression1">...</template>
3. <template [ng-switch-when] = "whenExpression1">...</template>
4. <template ng-switch-default>...</template>
5. </ANY>
- 6.

## ng-model

We can also use `ng-mobel` to bind a domain model to the form.

live demo ↗

```
1. @Component({
2.   selector: 'login-comp',
3.   directives: [FORM_DIRICTIVES],
4.   template: '<input type='text' [ng-form-control]='loginControl'
5.   [(ng-model)]='login'>'
6. })
7. class LoginComp {
8.   loginControl: Control = new Control('');
9.   login:string;
10. }
```

## CLASS DETAILS

NgSwitch

```
constructor(_validators:
  /* Array<Validator|Function> */ any[],
  _asyncValidators:
  /* Array<Validator|Function> */ any[],
  valueAccessors: ControlValueAccessor[])
```

## form : Control

## model : any

## viewModel : any

## onChanges(changes: {[key: string]: SimpleChange}) : void

## path : string[]



## LIBRARIES LEARN

Angular 2.0

5 Min

Google Group

Blog

Angular 1 for JS

Quickstart

Chat Room

Google+

Angular 1 for

Step by Step

Report an Issue

Twitter

Dart

Guide

GitHub

Angular

Full API

Material

Resources

AngularFire

Design Docs &amp;

Notes

## HELP

## COMMUNITY

Back to Top

SEARCH DOCS...

DOCS HOME

5 MIN QUICKSTART

TUTORIAL

DEVELOPER GUIDES

TESTING GUIDES

API PREVIEW

ANGULAR RESOURCES

HELP &amp; SUPPORT

# Injector

Class Stability: Stable

Angular 2 for TypeScript ▾

## WHAT IT DOES

A dependency injection container used for instantiating objects and resolving dependencies. An `Injector` is a replacement for a `new` operator, which can automatically resolve the constructor dependencies. In typical use, application code asks for the dependencies in the constructor and they are resolved by the `Injector`.

## HOW TO USE

```
import {Injector} from 'angular2/core';
...
class MyComponent {
  constructor(injector: Injector) {
    var someTypeInstance = injector.get(SomeType);
  }
}
```

## CLASS OVERVIEW

```
class Injector {iders)
  static resolveAndCreate
  static resolve(providers)

  constructor()

  get(token)
  getOptional(token)
  resolveAndCreateChild(providers)
  createChildFromResolved(providers)
  resolveAndInstantiate(provider)
  instantiateResolved(provider)
  displayName
  toString()
}
```

## CLASS DESCRIPTION

The following example creates an `Injector` configured to create `Engine` and `Car`.

live demo

```
1. @Injectable()
2. class Engine {
3. }
4.
5. @Injectable()
6. class Car {
7.   constructor(public engine:Engine) {}
8. }
9.
10. var injector = Injector.resolveAndCreate([Car, Engine]);
11. var car = injector.get(Car);
12. expect(car instanceof Car).toBe(true);
13. expect(car.engine instanceof Engine).toBe(true);
```

Notice, we don't use the `new` operator because we explicitly want to have the `Injector` resolve all of the object's dependencies automatically.

## CLASS DETAILS

`constructor(_proto: any, _parent?: Injector, _depProvider?: any, _debugContext?: Function)`

Private

`get(token: any) : any`

Retrieves an instance from the injector based on the provided token. Throws `NoProviderError` if not found.

live demo

```
1. var injector = Injector.resolveAndCreate([
2.   provide("validToken", {useValue: "Value"})
3. ]);
4. expect(injector.get("validToken")).toEqual("Value");
5. expect(() => injector.get("invalidToken")).toThrowError();
```

`Injector` returns itself when given `Injector` as a token.

```
1. var injector = Injector.resolveAndCreate([]);
2. expect(injector.get(Injector)).toBe(injector);
```

`getOptional(token: any) : any`

Retrieves an instance from the injector based on the provided token. Returns null if not found.

live demo

```
1. var injector = Injector.resolveAndCreate([
2.   provide("validToken", {useValue: "Value"})
3. ]);
4. expect(injector.getOptional("validToken")).toEqual("Value");
5. expect(() => injector.getOptional("invalidToken")).toBe(null);
```

`Injector` returns itself when given `Injector` as a token.

```
1. var injector = Injector.resolveAndCreate([]);
2. expect(injector.getOptional(Injector)).toBe(injector);
```

`parent : Injector`

Parent of this injector.

live demo

```
1. var parent = Injector.resolveAndCreate([]);
2. var child = parent.resolveAndCreateChild([]);
3. expect(child.parent).toBe(parent);
```

`resolveAndCreateChild(providers: Array<Type | Provider | any[]>) : Injector`

Resolves an array of providers and creates a child injector from those providers.

The passed-in providers can be an array of Type, `Provider`, or a recursive array of more providers.

live demo

```
1. class ParentProvider {}
2. class ChildProvider {}
3.
4. var parent = Injector.resolveAndCreate([ParentProvider]);
5. var child = parent.resolveAndCreateChild([ChildProvider]);
6.
7. expect(child.get(ParentProvider) instanceof ParentProvider);
8. expect(child.get(ChildProvider) instanceof ChildProvider);
9. expect(child.get(ParentProvider)).toBe(parent.get(ParentProvider));
```

This function is slower than the corresponding `createChildFromResolved` because it needs to resolve the passed-in providers first. See `Injector` and `Injector`.

`createChildFromResolved(providers: ResolvedProvider[]) : Injector`

Creates a child injector from previously resolved providers.

This API is the recommended way to construct injectors in performance-sensitive parts.

live demo

```
1. class ParentProvider {}
2. class ChildProvider {}
3.
4. var parent = Injector.resolveAndCreate([ParentProvider]);
5. var childProviders = Injector.resolve([ChildProvider]);
6.
7. var parent = Injector.fromResolvedProviders(parentProviders);
8. var child = parent.createChildFromResolved(childProviders);
9.
10. expect(child.get(ParentProvider) instanceof ParentProvider);
11. expect(child.get(ChildProvider) instanceof ChildProvider);
12. expect(child.get(ParentProvider)).toBe(parent.get(ParentProvider));
```

`resolveAndInstantiate(provider: Type | Provider) : any`

Resolves a provider and instantiates an object in the context of the injector.

The created object does not get cached by the injector.

live demo

```
1. @Injectable()
2. class Engine {
3. }
4.
5. @Injectable()
6. class Car {
7.   constructor(public engine:Engine) {}
8. }
9.
10. var injector = Injector.resolveAndCreate([Engine]);
11.
12. var car = injector.resolveAndInstantiate(Car);
13. expect(car.engine).toBe(injector.get(Engine));
14. expect(car).not.toBe(injector.resolveAndInstantiate(Car));
```

`resolveAndInstantiate(provider: ResolvedProvider) : any`

Instantiates an object using a resolved provider in the context of the injector.

The created object does not get cached by the injector.

live demo

```
1. @Injectable()
2. class Engine {
3. }
4.
5. @Injectable()
6. class Car {
7.   constructor(public engine:Engine) {}
8. }
9.
10. var injector = Injector.resolveAndCreate([Engine]);
11. var carProvider = Injector.resolve([Car])[0];
12. var car = injector.resolveAndInstantiate(carProvider);
13. expect(car.engine).toBe(injector.get(Engine));
14. expect(car).not.toBe(injector.resolveAndInstantiate(carProvider));
```

`displayName : string`

`toString() : string`

View Cheat Sheet

&lt; View Source Code

Improve this Doc

Back to Top



LIBRARIES LEARN

HELP

COMMUNITY

Angular 2.0

5 Min

Google Group

Blog

Angular 1 for JS

Quickstart

Chat Room

Google+

Angular 1 for Dart

Step by Step

Report an Issue

Twitter

Angular

Full API

GitHub

Material

Resources

AngularFire

Design Docs &amp;

Notes

## 2. Interaction

SEARCH DOCS...



DOCS HOME

5 MIN QUICKSTART

TUTORIAL

DEVELOPER GUIDES

TESTING GUIDES

API PREVIEW

ANGULAR RESOURCES

HELP &amp; SUPPORT

# NgSwitch

Directive

Angular 2 for TypeScript ▾

## WHAT IT DOES

The `NgSwitch` directive is used to conditionally swap DOM structure on your template based on a scope expression. Elements within `NgSwitch` but without `NgSwitchWhen` or `NgSwitchDefault` directives will be preserved at the location as specified in the template.

## HOW TO USE

```
<!-- platform directive - no import needed -->
<div [ng-switch]="conditionExpression">
  <template [ng-switch-when]="case1Exp">...</template>
  <template ng-switch-when="case2LiteralString">...</template>
  <template ng-switch-default>...</template>
</div>
```

## SELECTORS ?

[`ng-switch`]

View Cheat Sheet

&lt; &gt; View Source Code



Improve this Doc

SEARCH DOCS...

 DOCS HOME 5 MIN QUICKSTART TUTORIAL DEVELOPER GUIDES TESTING GUIDES API PREVIEW ANGULAR RESOURCES HELP & SUPPORT

Floating menu—below the fold

```
<template ng-switch-when="case1Exp">...</template>
<template ng-switch-when="case2LiteralString">...</template>
<template ng-switch-default>...</template>
</div>
```

SELECTORS 

[ng-switch]

OUTPUTS 

ng-model-change bound to NgFormControl.update

INPUTS 

ng-switch bound to NgSwitch.ngSwitch

EXPORTED AS 

form

## CLASS

class NgSwitch {

## OVERVIEW

ngSwitch

}

“Back to Top” fades in when the visible area is scrolled beneath the fold

## CLASS

NgSwitch simply chooses nested elements and makes them visible based on which element

## DESCRIPTION

matches the value obtained from the evaluated expression. In other words, you define a container element (where you place the directive), place an expression on the

[ng-switch-1=" " attribute] define any inner elements inside of the directive and place



View Cheat Sheet

 View Source Code 

Improve this Doc

 Back to Top

SEARCH DOCS...

 DOCS HOME 5 MIN QUICKSTART TUTORIAL DEVELOPER GUIDES TESTING GUIDES API PREVIEW ANGULAR RESOURCES HELP & SUPPORT

## Tooltip—hover over

```
<template ng-switch-when="case1Exp">...</template>
<template ng-switch-when="case2LiteralString">...</template>
<template ng-switch-default>...</template>
</div>
```

SELECTORS `[ng-switch]`

Tooltip text

OUTPUTS `ng-model-change` bound to `NgFormControl.update`INPUTS `ng-switch` bound to `NgSwitch.ngSwitch`EXPORTED AS `form`

## CLASS

```
class NgSwitch {
  ngSwitch
}
```

## OVERVIEW

## CLASS

`NgSwitch` simply chooses nested elements and makes them visible based on which element matches the value obtained from the evaluated expression. In other words, you define a container element (where you place the directive), place an expression on the

`[(ng-switch)]` attribute, define any inner elements inside of the directive and place

 View Cheat Sheet View Source Code Improve this Doc Back to Top

### 3. Spacing Guide

SEARCH DOCS...

DOCS HOME

5 MIN QUICKSTART

TUTORIAL

DEVELOPER GUIDES

TESTING GUIDES

API PREVIEW

ANGULAR RESOURCES

HELP &amp; SUPPORT

# NgSwitch

Directive

Stability: Stable

Angular 2 for TypeScript ▾

## WHAT IT DOES

The `NgSwitch` directive is used to conditionally swap DOM structure on your template based on a scope expression. Elements within `NgSwitch` but without `NgSwitchWhen` or

`NgSwitchDefault` directives will be preserved at the location as specified in the template.

## HOW TO USE

<!-- platform directive - no import needed -->

```
<div [ng-switch]="conditionExpression">
  <template [ng-switch-when]="case1Exp">...</template>
  <template ng-switch-when="case2LiteralString">...</template>
  <template ng-switch-default>...</template>
</div>
```

## SELECTORS ⓘ

[ng-switch]

## OUTPUTS ⓘ

`ng-model-change` bound to `NgFormControl.update`

## INPUTS ⓘ

`ng-switch` bound to `NgSwitch.ngSwitch`

## EXPORTED AS ⓘ

`form`

## CLASS

```
class NgSwitch {
  ngSwitch
}
```

## OVERVIEW

`NgSwitch` simply chooses nested elements and makes them visible based on which element matches the value obtained from the evaluated expression. In other words, you define a container element (where you place the directive), place an expression on the `[ng-switch] = "..."` attribute, define any inner elements inside of the directive and place a `[ng-switch-when]` attribute per element. The `when` attribute is used to inform `NgSwitch` which element to display when the expression is evaluated. If a matching expression is not found via a `when` attribute then an element with the `default` attribute is displayed.

live demo ↗

1. <ANY [ng-switch] = "expression">
2. <template [ng-switch-when] = "whenExpression1">...</template>
3. <template [ng-switch-when] = "whenExpression1">...</template>
4. <template ng-switch-default>...</template>
5. </ANY>
6. .

## ng-model

We can also use `ng-model` to bind a domain model to the form.

live demo ↗

```
1. @Component({
2.   selector: 'login-comp',
3.   directives: [FORM_DIRECTIVES],
4.   template: "<input type='text' [ng-form-control]='loginControl'
5.   [(ng-model)]='login'>"
6. })
7. class LoginComp {
8.   loginControl: Control = new Control('');
9.   login:string;
10. }
```

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## CLASS DETAILS

`NgSwitch`

```
constructor(_validators: /* Array<Validator|Function> */ any[], _asyncValidators: /* Array<Validator|Function> */ any[], valueAccessors: ControlValueAccessor[])
```

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## model : any

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## viewModel : any

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## onChanges(changes: {[key: string]: SimpleChange}) : void

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## path : string[]

Lore ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



View Cheat Sheet

View Source Code

Improve this Doc

Back to Top

LIBRARIES

LEARN

HELP

COMMUNITY

Angular 2.0

5 Min

Google Group

Blog

Angular 1 for JS

Quickstart

Chat Room

Google+

Angular 1 for

Step by Step

Report an Issue

Twitter

Dart

Guide

GitHub

Angular

Full API

Material

Resources

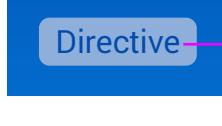
AngularFire

Design Docs &

Notes

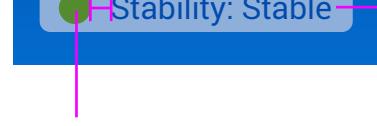
## 4. UI Components and Typography Guide

## API Type Label

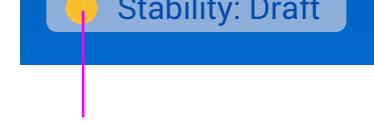


height:18px  
padding-left: 8px  
padding-right: 8px  
background-color: #8eafda  
border-radius: 4px;  
font: Roboto, regular, 12pt, #014bac

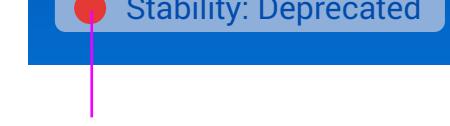
## Stability Label



width: 12px  
height: 12px  
border-radius: 6px  
background-color: #558b2f



width: 12px  
height: 12px  
border-radius: 6px  
background-color: #FBC02D

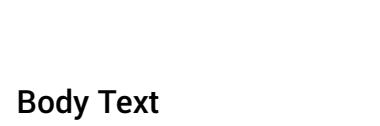


width: 12px  
height: 12px  
border-radius: 6px  
background-color: #E53935

## Title

WHAT IT DOES font: Roboto, regular, 16pt, #78909c

## Tooltip Icon



8px

SELECTORS ?

width: 16px  
height: 16px  
color: #78909c  
url download: https://www.google.com/design/icons/#ic\_help

## Tooltip Icon—hover state



Tooltip text

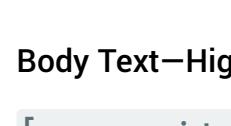
height: 22px  
color: #78909c  
opacity: 90%  
border-radius: 2px  
font: Roboto, Medium, 10px, #ffffff

## Body Text

The `NgSwitch` directive is used to conditionally swap DOM structure on your template based on a scope expression. Elements within `NgSwitch` but without `NgSwitchWhen` or `NgSwitchDefault` directives will be preserved at the location as specified in the template.

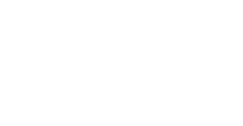
font: Roboto, regular, 16pt, #5c7078  
line-height: 28pt

## Body Text—Link



font: Roboto Mono, regular, 16pt, #2376ca  
background-color: #eef1f2  
border-radius: 2px  
border-bottom: 0.5px, dotted, #2376ca

## Body Text—Link, Hover



border-bottom: 0.5px, solid, #2376ca

## Body Text—Highlight without linking

[ng-switch] background-color: #eef1f2  
border-radius: 2px

## Body Text—Code

```
<!-- platform directive - no import needed -->
<div [ng-switch]="conditionExpression">
  <template [ng-switch-when]="case1Exp">...</template>
  <template ng-switch-when="case2LiteralString">...</template>
  <template ng-switch-default>...</template>
</div>
```

## Regular text

font: Roboto Mono, regular, 16pt, #5c7078  
line-height: 28pt

## Highlight 1

font: Roboto Mono, regular, 16pt, #9e9d24

## Highlight 2

font: Roboto Mono, regular, 16pt, #d43669

## Secondary punctuation - {}();

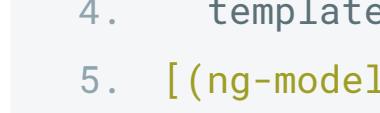
## Code comments

font: Roboto Mono, regular, 16pt, #90a4ae

## Sub-title

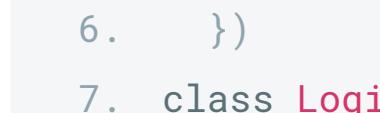
ng-model font: Roboto, regular, 20pt, #5c7078

## Code Snippet



background-color: #cf8dc  
height: 36px  
The top-left and top-right round corner: 4px

live demo ↗

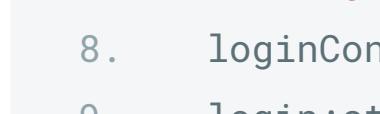


font: Roboto, regular, 14pt, #2376ca  
border-bottom: 0.5px, dotted, #2376ca

Hover state border-bottom: 0.5px, solid, #2376ca

8px

width: 14px  
height: 14px  
color: #2376ca  
url download: https://www.google.com/design/icons/#ic\_exit\_to\_app



background-color: #f5f6f7  
padding: 16px  
The bottom-left and bottom-right round corner: 4px

## Regular text

font: Roboto Mono, regular, 14pt, #5c7078  
line-height: 26pt

## Highlight 1

font: Roboto Mono, regular, 14pt, #9e9d24

## Highlight 2

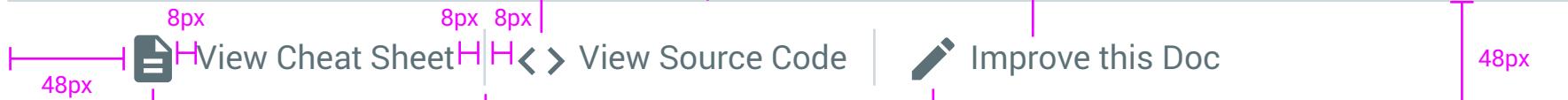
font: Roboto Mono, regular, 14pt, #d43669

## Secondary punctuation - {}();

## Code comments

font: Roboto Mono, regular, 14pt, #90a4ae

## Horizontal rule



1px, solid, #ecefef

## Floating menu



width: 24px  
height: 24px  
color: #5c7078  
url download:  
https://www.google.com/design/icons/#ic\_description

url download: https://www.google.com/design/icons/#ic\_code

1px, solid, #cf8dc

8px

8px

8px

8px

8px

8px

8px

8px

font: Roboto, regular, 14pt, #5c7078

url download: https://www.google.com/design/icons/#ic\_create

url download: https://www.google.com/design/icons/#ic\_improve

48px

48px

48px

48px

48px

48px

48px

48px

↑ Back to Top ↗

url download:

https://www.google.com/design/icons/#ic\_arrow\_upward