# Roadmap Angular

## De zéro à héros 

CodeWise

formations pour développeur web

# Sommaire

# Introduction

## Qu’est-ce qu’une roadmap?

Une roadmap est une représentation graphique simplifiée permettant de suivre un fil conducteur pour atteindre un objectif.

## Comment utiliser cette roadmap ?

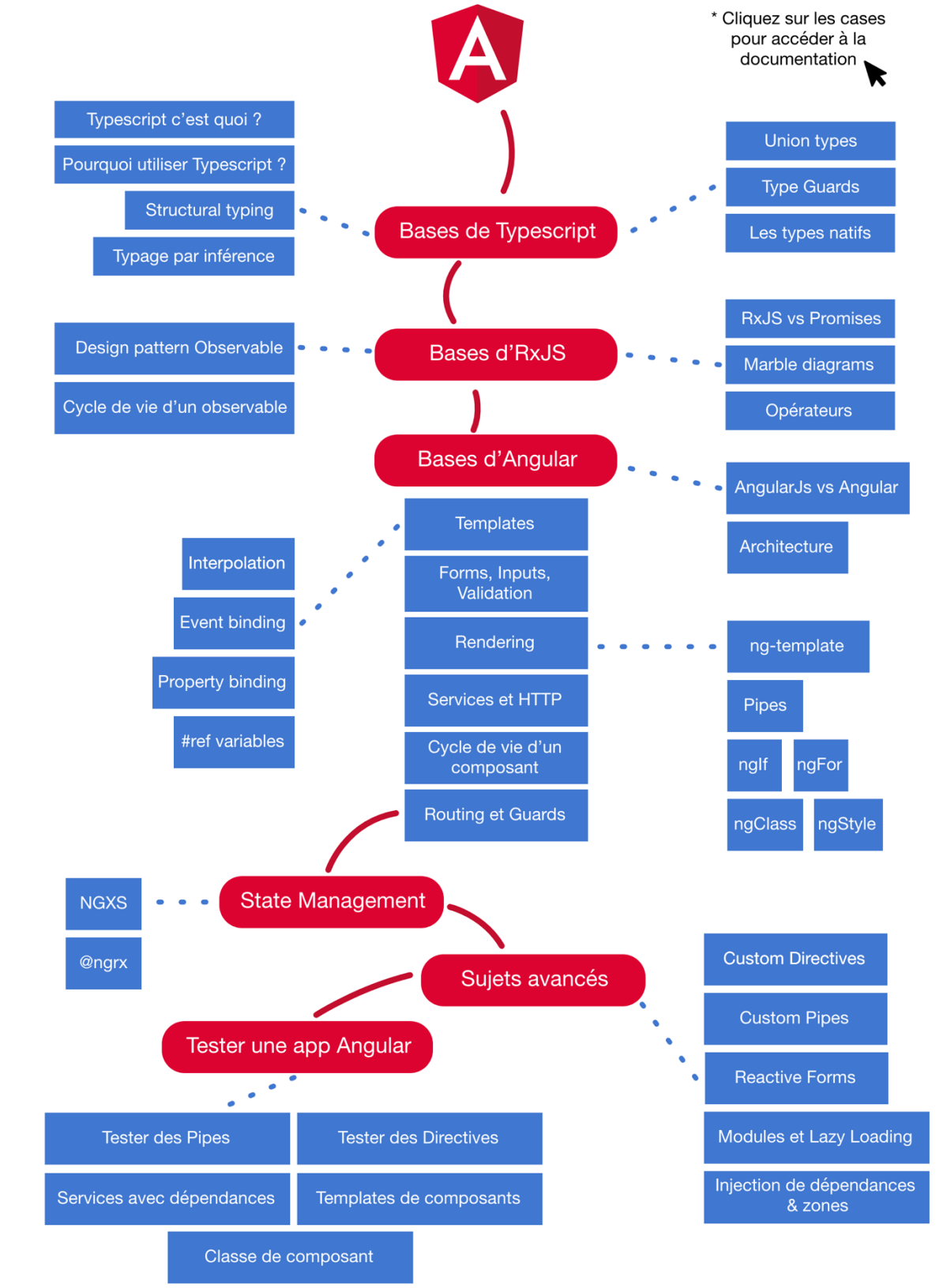
Cette roadmap a pour objectif de vous représenter tout ce que vous devriez connaître sur le framework Angular, des sujets basiques aux plus avancés.

Elle peut vous servir à repérer où vous en êtes dans votre parcours de développeur Angular.

Vous pouvez cliquer sur les cases pour accéder à des documentations ou articles en lien avec le sujet. *

À la suite de cette roadmap, vous trouverez un dictionnaire contenant toutes les définitions des termes présents sur la roadmap.

# Roadmap


* Cliquez sur les cases pour accéder à la documentation
Typescript c'est quoi ?
Pourquoi utiliser Typescript ?
Structural typing
Typage par inférence
Bases de Typescript
Union types
Type Guards
Les types natifs
Design pattern Observable
Cycle de vie d'un observable
Bases d'RxJS
RxJS vs Promises
Marble diagrams
Opérateurs
Bases d'Angular
AngularJs vs Angular
Architecture
Templates
Forms, Inputs, Validation
Rendering
Services et HTTP
Cycle de vie d'un composant
Routing et Guards
Interpolation
Event binding
Property binding
#ref variables
ng-template
Pipes
ngIf
ngFor
ngClass
ngStyle
NGXS
@ngrx
State Management
Sujets avancés
Custom Directives
Custom Pipes
Reactive Forms
Modules et Lazy Loading
Injection de dépendances & zones
Tester une app Angular
Tester des Pipes
Tester des Directives
Services avec dépendances
Templates de composants
Classe de composant

# Lexique

## Typescript

Typescript est un langage de programmation développé par Microsoft, il s'agit d'un sur-ensemble de Javascript, il a pour but d'améliorer et de sécuriser la production de code Javascript.

https://www.typescriptlang.org/

**Les types natifs** : Typescript propose un ensemble de types courants intégrés de manière native. (string, boolean, number etc.)

Lien vers la documentation

**Union Types** : Un type d'union est un type formé de deux ou plusieurs autres types.

Lien vers la documentation

**Type Guards** : Un type guard est une expression qui effectue une vérification d'exécution qui garantit le type dans une certaine étendue.

Lien vers la documentation

**Structural typing** : Le structural typing est une manière de concevoir les types. On considère que deux objets sont de même type s'ils possèdent les mêmes propriétés.

Lien vers la documentation

**Typage par inférence** : Le typage par inférence est utilisé pour typer des variables lorsqu'il n'y a pas d'annotation de typage explicite.

Lien vers la documentation

# RxJS

RxJS est une bibliothèque de programmation réactive utilisant les observables, pour faciliter la composition de code asynchrone.

https://rxjs.dev/

**Design pattern Observer** : Les observables RxJS sont une implémentation du design pattern Observer. Ce design pattern fournit une méthode uniforme permettant de définir un rapport de dépendance entre deux objets ou plus afin de communiquer des modifications à un autre objet de façon aussi simple et rapide que possible.

Lien vers un article lié

**Cycle de vie d'un observable** :

Lien vers un article lié

**RxJS vs Promises** : Les promesses sont une alternative aux observable, pour gérer la programmation asynchrone.

Tableau comparatif

Lien vers un article lié

**Marble diagrams** : Ce sont des diagrammes qui vous fournissent une représentation visuelle d'un observable. Vous pouvez les utiliser pour tester le comportement de différents observables et opérateurs.

Lien vers la documentation

**Opérateurs** : Les opérateurs sont des éléments essentiels d'RxJS qui permettent de composer facilement du code asynchrone complexe de manière déclarative.

Lien vers la documentation

## Angular - https://angular.io/

Angular est un framework côté client, open source, basé sur Typescript, et co-dirigé par l'équipe du projet « Angular » de Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète d'AngularJS, construit par la même équipe.

**AngularJs vs Angular** : AngularJs est l'ancienne version d'Angular, elle n'est plus très utilisée aujourd'hui.

La différence la plus fondamentale entre les deux frameworks open-source est qu'Angular est basé sur Typescript alors que AngularJS est basé sur Javascript.

Lien vers un article lié

**Architecture** : L'architecture d'Angular est simplement un arbre hiérarchique de composants. Ces composants sont réutilisables et indépendants rendant le code Angular hautement testable. Ils interagissent avec des services, qui servent à gérer la logique non-graphique de l'application, et forment ainsi un pattern MVVM (Model - View - ViewModel)

Lien vers un article lié

**Forms, Inputs, Validation** : Angular propose deux approches différentes pour gérer les entrées des utilisateurs via des formulaires : Les reactive forms et les template-driven forms. Commencez par voir les template-driven forms si vous débutez.

Lien vers la documentation

**Services et Http** : Les services permettent de réutiliser du code entre différents composants, de faciliter l'échange de données, de séparer les responsabilités visuelles et fonctionnelles. Ils servent à gérer toute la logique métier de l'application, ou plus généralement tout ce qui n'est pas directement de la gestion d'interface utilisateur.

Lien vers un article lié

**Cycle de vie d'un composant** : Le cycle de vie correspond aux différentes étapes de la vie d'une instance. Il débute toujours par une instanciation et termine par la destruction de l'instance, donc la désallocation des ressources mémoire qui lui sont affectées.

Angular nous permet d'agir sur le cycle de vie des composants et des directives.

Lien vers un article lié

**Routing et Guards** :  Dans une application Angular (SPA), vous modifiez ce que l'utilisateur voit en affichant ou en masquant dynamiquement des parties de l'affichage (composants), en restant toujours sur la même page.

Pour recréer l'illusion de naviguer entre plusieurs pages, vous devez utiliser le routeur d'Angular.

Lien vers la documentation

**Interpolation** : L'interpolation vous permet d'incorporer des valeurs de chaîne dynamiques dans vos modèles HTML. Utilisez l'interpolation pour modifier dynamiquement ce qui apparaît dans une vue d'application, par exemple en affichant un message d'accueil personnalisé qui inclut le nom de l'utilisateur.

Lien vers un article lié

**Pipes** : Les Pipes sont des filtres utilisables directement depuis la vue afin de transformer les valeurs lors du "binding".

Lien vers un article lié

**Event binding** : Permet d'exécuter une fonction dans un composant en réaction à un événement émis par un élément du DOM.

Lien vers un article lié

**Property binding** : Permet de valoriser une propriété d'un composant ou d'une directive à partir du modèle, si un changement est intervenu sur une valeur de ce dernier.

Lien vers un article lié

**Directives ngIf, ngFor, ngStyle, ngClass** :

- Les directives structurelles : Elles ont pour but de modifier le DOM en ajoutant, enlevant ou remplaçant un élément du DOM.
- Les directives d'attribut : Elles ont pour but de modifier l'apparence ou le comportement d'un élément.

Lien vers un article lié

**Ng-template** : Ng-template est une directive qui permet de gérer le rendering des éléments du DOM. Elle permet d'afficher/dupliquer/masquer l'élément en fonction des conditions.

Les directives structurelles sont d'ailleurs basées sur cette directive <ng-template>

Lien vers un article lié

**#ref variables** : Les variables de template vous aident à utiliser les données d'un template dans une autre partie du template. Utilisez les variables de template pour effectuer des tâches telles que la réponse aux entrées de l'utilisateur ou le réglage fin des formulaires de votre application.

Lien vers un article lié

## State management

Utile pour gérer des applications Angular qui grandissent, les systèmes de State Management servent à stocker des états de manière globale.

**ngxs** : ngxs est un modèle de gestion d'état + une bibliothèque pour Angular. Il agit comme une source unique de vérité pour l'état de votre application.

Lien vers un article lié

**ngrx** : ngrx est un groupe de librairies inspirées par le pattern Redux, qui est lui-même inspiré du pattern Flux. Le pattern ngrx est un pattern de gestion d'état. Il stocke l'état de l'application (State) dans un Store.

Lien vers un article lié

## Sujets avancés

**Custom directives** : Lorsque aucune directive native correspond à votre besoin, vous pouvez créer vos propres directives.

Lien vers un article lié

**Custom pipes** : De la même manière, si aucun pipe natif correspond à votre besoin, vous pouvez créer vos propres pipes.

Lien vers un article lié

**Reactive forms** : Les reactive forms permettent un contrôle plus fin que les template-driven forms. C'est très pratique pour la validation et autres opérations que vous serez amené à effectuer lors du développement d'une véritable application.

Lien vers la documentation

Lien vers un article lié

**Module et lazy loading** : Le lazy loading permet de charger les modules à la demande afin de ne pas gêner le chargement initial de l'application. Utile pour améliorer les performances des applications publiques et le référencement.

Lien vers un article lié

**Injection de dépendances et zones** : Quand Angular instancie un composant, il peut résoudre les dépendances en les injectant via le constructeur du composant.

Une zone est un contexte d'exécution qui persiste pendant les tâches asynchrones.

Vous pouvez les voir comme un stockage local pour les threads de votre VM JavaScript

Lien vers la documentation

Lien vers un article lié