Mettez à jour vos applications avec Angular5 !

Sous-titre de l'article

Synopsis de l'article

Texte affiché avant le sommaire.

Multipage.

Texte de la licence de remplacement Référence de la licence de remplacement

Niveau du lecteur Durée de lecture

Liens supplémentaires.

# Introduction

Vous êtes sûrement au courant, Angular 5 est sorti le 1er Novembre 2017 ! Concrètement, cela signifie qu’il est maintenant temps de voir ce que cette nouvelle version du Framework propose, et si nous pouvons en profiter dans nos applications. Allez, au boulot…

La version précédente était la version ~4.4.x, et l’équipe en charge du développement d’Angular a décidé qu’il était temps de passer à la vitesse supérieure, littéralement. En effet, l’objectif principal de cette mise à jour est de **rendre Angular plus rapide**.

Dans cet article nous allons passer en revue les **12 principaux changements** apportés par la version 5 d’Angular. Mais avant, nous allons voir comment vous pouvez déjà mettre à jour votre application grâce à un outil très pratique que Google nous propose.

|  |  |
| --- | --- |
| info Bordure Fond | |
| [KitImage] | Remarque : Saviez-vous que les plateformes comme Google AdWords ou Adsense utilisent Angular pour construire leur interface ? |

# Mettre à jour son application pour Angular 5

Alors, bonne nouvelle, l'équipe d'Angular ne nous a pas laissé seul pour mettre à jour nos applications ! Elle a même construit un outil vraiment génial pour rendre la mise à niveau aussi facile que possible.

Le fonctionnement de cet outil est simple :

* vous entrez la version actuelle de votre application ;
* vous entrez la version vers laquelle vous souhaitez adapter votre application. Dans notre cas il s'agit de la version 5 ;
* ensuite, vous suivez les instructions recommandées par l'outil.

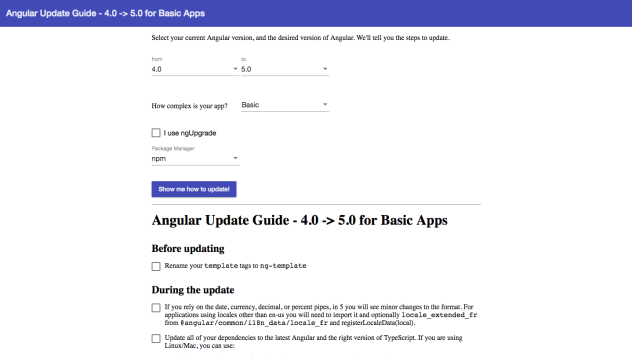


Figure 1 - L'outil proposé par Google pour mettre à jour vos applications.

## Exemple d’adaptation de la version 4.3

Prenons le cas d’un cours écrit sur Angular que je dois mettre à jour. Comment s’y prendre ? Je pensais passer des jours à relire tout le contenu du cours, revoir tel ou tel chapitre en fonction des dernières nouveautés, etc. En gros, une galère ! Mais grâce à l'outil présenté ci-dessus, la mise à jour va pouvoir être faite rapidement. L'outil m'informe que **je dois suivre seulement quatre étapes pour mettre à jour mon cours** :

* renommer mes balises *template* en *ng-template*. Comme il n'y pas de balise de ce type dans mon tutoriel, je peux passer cette étape ;
* il ne faut plus utiliser le module *HttpModule* et le service *Http*, mais le module *HttpClientModule* et le service *Http*. Je vais donc devoir mettre à jour le chapitre sur Http, tant pis ! L'application m'informe que le service *HttpClient* simplifie le traitement des requêtes, et que je n'aurai plus besoin d'appeler la fonction *map* pour extraire le *json* de la réponse. Ce nouveau service prend également en charge les valeurs de retour typées et les intercepteurs ;
* apparemment, il y a quelques changements au niveau de certains pipes natifs, concernant l'internationalisation. Le cours n'est donc pas concerné par ces changements, ouf ! On peut passer à l'étape suivante ;
* pour terminer, il faut mettre à jour toutes les dépendances vers la dernière version d'Angular, et la bonne version de TypeScript. L'application nous facilite la tâche en nous donnant directement les commandes à appliquer.

|  |
| --- |
| Titre  shell Oui Ligne de début.Dissimulable Lien fichier |
| npm install @angular/{animations,common,compiler,compiler-cli,core,forms,http,platform-browser,platform-browser-dynamic,platform-server,router}@'^5.0.0' typescript@2.4.2 rxjs@'^5.5.2' --save    npm install typescript@2.4.2 --save-exact |

|  |  |
| --- | --- |
| warning Bordure Fond | |
| [KitImage] | Angular 5 nécessite d’installer au moins la version 2.4.2 de TypeScript. |

J’ai pris la liberté d’ajouter l’option *--save* à la première commande, pour mettre directement à jour le fichier de déclaration des dépendances *package.json*.

Bilan, je vais devoir modifier une partie du chapitre « *Hello, world* » de mon cours pour être à jour avec ces nouvelles dépendances, ainsi que mettre à jour le chapitre « *Http* » pour utiliser le nouveau module *HttpClientModule*. Mais je m’en sors plutôt bien, seulement deux chapitres à réécrire !

Bref, je n’ai plus qu’à suivre ces instructions pour mettre mon cours à jour. Et pour votre application, je vous invite à suivre la même procédure, en fonction de la version actuelle que vous utilisez. Pratique non ?

Pour les plus téméraires, vous retrouverez ci-dessous les principaux changements apportés par cette nouvelle version d’Angular.

# Les 12 travaux … de Angular 5

Voici une liste des changements qu’apporte Angular 5, résumée en 12 points.

Il est nécessaire d’avoir de bonnes connaissances sur Angular (animations, Angular Universal, etc.) pour profiter pleinement de ces explications, sinon vous pouvez simplement utiliser l’outil ci-dessus, qui vous suffira dans de nombreux cas.



Figure 2 - Les 12 travaux d'Hercule ... d'Angular 5

## Le module *HttpClient* remplace (définitivement) *Http*

Avant la version d’Angular 4.3, c’est le module *@angular/http* qui était utilisé pour gérer les requêtes Http. Mais dans la version 5, l’équipe en charge d’Angular a décidé de déprécier le module *Http*. À la place, c’est le service *HttpClient*, du package *@angular/common/http* (apparut dans la version 4.3) qui est maintenant recommandé.

Les fonctionnalités de *HttpClient* diffèrent de *Http*. Voici un [post sur Stackoverflow](https://stackoverflow.com/questions/45129790/difference-between-http-and-httpclient-in-angular-4) qui résume très bien la différence entre ces deux modules. Quant aux importations, elles sont aussi différentes.

Le *Http* (avant) :

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| import { HttpModule } from '@angular/http'; |

Et le *HttpClient* (après) :

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| import { HttpClientModule } from '@angular/common/http'; |

## Le support pour plusieurs alias d’exportation

Dans Angular 5, vous pouvez maintenant donner plusieurs noms à vos composants et directives lors de l’exportation. L’exportation d’un composant avec plusieurs noms peut vous aider à maintenir votre code plus facilement. Si vous devez changer les noms de vos composants par exemple, et bien vous pourrez toujours utiliser l’ancien nom dans le reste de l’application.

Bon, pratique, mais pas indispensable, on est d’accord.

*Exemple d’utilisation :*

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| import { Component } from '@angular/core';    @Component({   selector: 'app-root',   templateUrl: './app.component.html',   styleUrls: ['./app.component.css'],   exportAs:'dashboard, logBoard' // on ajoute plusieurs noms  })  export class AppComponent {    title = 'app';  } |

## Les pipes *number*, *date* et *currency* sont internationalisés

Angular 5 dispose d’une nouvelle version des pipes : *number*, *date* et *currency*. Ils améliorent la standardisation entre les navigateurs, et éliminent le besoin des polyfills pour i18n. Les pipes dépendent du [CLDR](http://cldr.unicode.org/) (Unicode Common Locale Data Repository, rien que ça !) pour s’adapter en fonction de la langue que vous souhaitez prendre en charge.

**Remarque** : Si, pour une raison obscure, vous souhaitez utiliser les anciens Pipes, c’est encore possible. Vous devrez importer le *DeprecatedI18NPipesModule* après le *CommonModule* :

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| import { NgModule } from '@angular/core';  import { CommonModule, DeprecatedI18NPipesModule } from '@angular/common';    @NgModule({    imports: [      CommonModule,      // importez le module déprécié après...      DeprecatedI18NPipesModule    ]  })  export class AppModule { } |

## Support des décorateurs améliorés

Angular 5 supporte l’utilisation des expressions lambda (ou fonctions fléchées), à la place des fonctions nommées, notamment pour définir la valeur de *useValue*, *useFactory* et *data*.

*Avec Angular 5 :*

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| @Component({  provider: [{provide: 'token', useFactory: () => null}]  })  export class AwesomeClass {} |

*Avant Angular 5 :*

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| @Component({  provider: [{provide: 'token', useValue: calculated()}]  })  export class AwesomeClass {} |

## Optimisation du processus de *build*

L’équipe Angular s’est concentrée principalement à rendre Angular 5 plus rapide, plus léger et plus facile à utiliser. Dans Angular 5, les *builds* de production créés avec Angular CLI appliquent désormais le *build optimizer* par défaut.

Le *build optimizer* supprime les décorateurs Angular des codes d’exécution de votre application, réduisant ainsi la taille de votre *build*, et augmentant la vitesse de démarrage de votre application.

De plus, le *build optimizer* supprime une partie de votre application qui n’est pas nécessaire lors de l’exécution. Cette action entraîne une réduction de la taille du *build* et une vitesse d’application plus rapide. Que du bonus !

## Une API Transfer pour Angular Universal

Si vous n’utilisez pas Angular Universal pour le moment, comme moi, passez directement au point suivant. Cependant, pour les autres, voici ce que vous devez retenir :

* l’équipe d’Angular Universal a ajouté [Domino](https://github.com/fgnass/domino) à la *platform-server*. Cela signifie simplement qu’il y a plus de manipulation du DOM qui peuvent se produire en dehors des contextes côté serveur ;
* en outre, deux modules, *ServerTransferStateModule* et *BrowserTransferModule* ont été ajoutés à Angular Universal. Ces modules vous permettent de générer des informations dans le cadre de votre rendu avec *platform-server*, puis de les transférer vers le client pour éviter la génération de la même information. En résumé, il transfère l’état du serveur, ce qui signifie que les développeurs n’ont pas besoin d’effectuer une seconde requête HTTP une fois que l’application a accédé au client.

## Une compilation beaucoup plus rapide

De nombreuses améliorations ont été apportées au compilateur Angular pour améliorer sa rapidité. Le compilateur Angular profite maintenant des transformations de TypeScript. Les transformations TypeScript sont une nouvelle fonctionnalité introduite dans TypeScript 2.3, qui permet de se connecter au pipeline de compilation standard de TypeScript. C’est un peu technique, mais retenez que ça va plus vite.

Vous pouvez en profiter en exécutant :

|  |
| --- |
| Titre  shell Oui Ligne de début.Dissimulable Lien fichier |
| ng serve --aot |

La documentation officielle [angular.io](http://angular.io/) elle-même a été utilisée comme étude de cas, et le pipeline du compilateur a permis d’économiser 95% du temps de *build* !

Concrètement, on passe d’une minute pour le temps de *build* à quelques secondes. Angular 5 propose donc des avancées plutôt solides.

## La validation des formulaires dans Angular 5

Dans Angular 5, les formulaires ont désormais la possibilité de décider quand la validité et la valeur d’un champ ou d’un formulaire seront mises à jour, via *blur* ou *submit*. Cela permet d’affiner l’ancien comportement qui se déclenche à chaque événement d’entrée.

*Exemple pour un champ :*

|  |
| --- |
| Titre  html Oui Ligne de début.Dissimulable Lien fichier |
| <input name="nickName" ngModel [ngModelOptions]="{updateOn: 'blur'}"> |

*Exemple pour un formulaire :*

|  |
| --- |
| Titre  html Oui Ligne de début.Dissimulable Lien fichier |
| <form [ngFormOptions]="{updateOn: 'submit'}"> |

Dans le cas des *Reactive forms*, on peut ajouter l’option comme ceci, côté composant :

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| ngOnInit() {  this.newUserForm = this.fb.group({  userName: ['Bob', { updateOn: 'blur', validators: [Validators.required] }]  });  } |

## Les animations avec Angular 5

Dans Angular 5, nous avons désormais deux nouveaux alias de transition. Il s’agit de :*increment* et :*decrement*.

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| ...  animations: [  trigger('bannerAnimation', [  transition(":increment", group([  query(':enter', [  style({ left: '100%' }),  animate('0.5s ease-out', style('\*'))  ]),  query(':leave', [  animate('0.5s ease-out', style({ left: '-100%' }))  ])  ])),  transition(":decrement", group([  query(':enter', [  style({ left: '-100%' }),  animate('0.5s ease-out', style('\*'))  ]),  query(':leave', [  animate('0.5s ease-out', style({ left: '100%' }))  ])  ])),  ])  ] |

Les requêtes d’animation prennent désormais en charge les limites négatives, auquel cas les éléments sont appariés depuis la fin, plutôt que depuis le début, comme ceci:

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| ... animations: [  trigger( 'myAnimation', [  transition( '\* => go', [  query( '.item', [  style({opacity: 0}),  animate('1s', style({opacity: 1})),  ],  {limit: -3}),  ]),  ]),  ] ... |

Enfin, si vous n’avez pas encore vu les animations avec Angular, cela ne doit pas vous parler beaucoup.

(Dans ce cas, utilisez l’outil que je présente au début du chapitre, ce sera plus simple et plus rapide.)

## Les nouveaux événements de cycle de vie du Router

Certains nouveaux événements de cycle de vie ont été ajoutés au routeur. Voici la liste complète de ces événements :

* *GuardsCheckStart*
* *ChildActivationStart*
* *ActivationStart*
* *GuardsCheckEnd*
* *ResolveStart*
* *ResolveEnd*
* *ActivationEnd*
* *ChildActivationEnd*

Avec ces événements, les développeurs peuvent suivre le cycle du routeur depuis le début des Guards en cours, jusqu’à la fin de l’activation.

De plus, vous pouvez désormais configurer le routeur pour recharger une page, même lorsqu’il reçoit une requête pour naviguer vers une URL identique :

|  |
| --- |
| Titre  typescript Oui Ligne de début.Dissimulable Lien fichier |
| providers: [  // ...  RouterModule.forRoot(routes, {  onSameUrlNavigation: 'reload'  })  ] |

## Meilleur support des *Service Workers*

Dans Angular 5, nous avons un meilleur support pour les *services workers* via le packet [@angular/service-worker](https://github.com/angular/angular/tree/master/packages/service-worker).

Le package de *service worker* est un dérivé conceptuel du package *@angular/service-worker* qui a été conservé sur : github.com/angular/mobile-toolkit. Mais il a été réécrit pour prendre en charge des cas d’utilisation sur une plus grande variété d’applications.

|  |  |
| --- | --- |
| info Bordure Fond | |
| [KitImage] | Pour le moment, vous devez intégrer manuellement ce package, car il n’est pas pleinement intégré avec Angular CLI v1.5. Il est disponible en bêta dans la version 1.6 du CLI. |

## Déprécations, et mises à jour diverses

* *NgFor* a été retiré et est déprécié depuis la version 4. Utilisez *NgForOf* à la place. Dans tous les cas, cela n’a aucune incidence sur l’utilisation de *\*ngFor*dans vos templates.
* L’option du compilateur *enableLegacyTemplate* est maintenant désactivée par défaut, car la balise *<template>* était obsolète depuis la version 4. Utilisez *<ng-template>* à la place. L’option *enableLegacyTemplate* et l’ancienne balise seront toutes deux supprimées dans Angular 6. Simple anticipation !
* La méthode *ngGetSelectors* a été retirée et dépréciée depuis la version 4. Utilisez *ComponentFactory.ngContentSelectors* à la place.
* *ReflectiveInjector* est maintenant déprécié. Utilisez *Injector.create* en remplacement.
* *NgProbeToken* est maintenant retiré du *@angular/platform-browser* et est déprécié depuis la version 4. Importez le depuis *@angular/core* à la place.

# Conclusion

J’espère que ce modeste article vous aura donné envie de mettre à jour votre application, et ce sans prise de tête.

Êtes-vous déjà passé à Angular 5 ? Avez-vous remarqué une amélioration significative ? Faites-moi savoir dans les commentaires !

Et pour terminer, voici un lien vers les [derniers correctifs d’Angular](https://github.com/angular/angular/blob/master/CHANGELOG.md), mis à jour régulièrement par l’équipe Angular elle-même.

## Ressources

* Outil pour la mise à jour : https://angular-update-guide.firebaseapp.com/.
* Angular Changelog : <https://github.com/angular/angular/blob/master/CHANGELOG.md>.
* Site CLDR : <http://cldr.unicode.org/>.

## Remerciements

<Insérer les remerciements pour la relecture technique et orthographique>

## Pour aller plus loin

Cet article présente les nouveautés de la version 5 d’Angular, mais il y en aura d’autres, tous les six mois exactement ! Pour être tenu informé de la prochaine mise à jour d’Angular, je ne peux que vous invitez à me suivre sur mon blog : <https://awesome-angular.com>. Merci à vous pour votre lecture et vos retours éventuels, et à très vite !