# [Angular : JiT vs AoT](https://sebastienollivier.fr/blog/angular/angular-jit-vs-aot)

[Publié le 03 octobre 2017](https://sebastienollivier.fr/blog/angular/angular-jit-vs-aot)

Angular propose deux types de compilation de templates : JiT (Just-in-Time) et AoT (Ahead-of-Time). Pourquoi ? Quelles sont les différences ? Lequel dois-je utiliser ? Nous allons voir cela ensemble dans cet article.

## Angular compiler

Une application Angular est composée de fichiers TypeScript (components, directives, pipes, etc.) et de templates (inline ou sous forme de fichiers HTML). Angular utilise ces templates pour déterminer la façon dont va se faire l'affichage des composants.

Mais Angular ne les utilise pas tels quels. Il va les compiler ([transpiler](https://en.wikipedia.org/wiki/Source-to-source_compiler) pour être plus exact) : un template va engendrer une fonction Javascript. C'est cette fonction qui sera ensuite appelée lorsque Angular aura besoin de générer la vue associée au template. De cette manière, le framework n'est pas obligé de parser le HTML à chaque fois. Angular, dans ses sources, intègre un compilateur (angular compiler) qui va être responsable d'effectuer ce traitement.

## JiT : Just-in-Time

Par défaut, la compilation des templates d'une application va être effectuée pendant l'exécution de l'application, c'est à dire que Angular va compiler à la volée les templates. C'est ce qu'on appelle la compilation JiT qui est utilisée via la commande ng build ou ng build --prod --no-aot.

Dans cette configuration, le bundle JavaScript généré va intégrer les templates de l'application sous une syntaxe HTML. Si vous inspectez le fichier main.bundle.js généré par la build, vous trouverez des portions de code contenant vos templates, comme ci-dessous :

/\*\*\*/ }),

/\*\*\*/ "../../../../../src/app/components/home/home.component.html":

/\*\*\*/ (function(module, exports) {

module.exports = "{{user|json}}\n\n&lt;button (click)='logoff()'&gt;logoff&lt;/button&gt;"

/\*\*\*/ }),

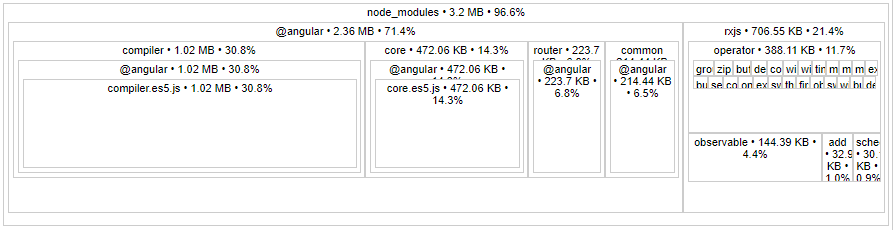
A l'exécution, le compiler Angular va prendre ces templates pour les transformer en fonction Javascript.

Ce mécanisme a deux effets négatifs. Le premier est que le bundle Javascript va être plus gros puisque les sources de l'application devront intégrer le compilateur de templates (dans le fichier vendor.bundle.js). Le second point est que l'application va devoir compiler les templates lors de son exécution ce qui aura nécessairement un impact négatif sur le temps d'affichage.

Pour donner un ordre d'idée, sur une application simple, la compilation JiT donne les bundles suivants :

* main.bundle.js : 63k (21k en minifié)
* vendor.bundle.js : 3321k (960k en minifié)

Une analyse du fichier vendor.bundle.js (en utilisant [source-map-explorer](https://github.com/danvk/source-map-explorer/blob/master/README.md)) montre que le compiler Angular prend 35% de la taille du bundle :



## AoT : Ahead-of-Time

La compilation des templates est invariante, c'est-à-dire qu'un même template engendrera toujours la même fonction (de la même manière que la compilation TypeScript par exemple). Cela veut dire que la compilation des templates peut faire partie du process de build, puisqu'elle n'est pas dépendante de son contexte d'utilisation.

C'est sur cette constatation que se base le principe de la compilation AoT, utilisable via les commandes ng build --aot ou ng build --prod. Plutôt que de compiler les templates au lancement de l'application, cette compilation va être effectuée lors de la phase de build, directement par webpack. Le bundle généré ne contiendra plus les templates HTML mais directement les templates compilés. Si vous inspectez le fichier main.bundle.js généré par la build, vous trouverez des portions de code contenant vos templates compilés, comme ci-dessous :

function View\_HomeComponent\_0(\_l) {

return \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_30" /\* ɵvid \*/](0, [(\_l()(), \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_28" /\* ɵted \*/](null, ['', '\n\n'])), \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_26" /\* ɵpid \*/](0, \_\_WEBPACK\_IMPORTED\_MODULE\_2\_\_angular\_common\_\_["e" /\* JsonPipe \*/], []), (\_l()(), \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_14" /\* ɵeld \*/](0, null, null, 1, 'button', [], null, [[null, 'click']], function (\_v, en, $event) {

var ad = true;

var \_co = \_v.component;

if (('click' === en)) {

var pd\_0 = (\_co.logoff() !== false);

ad = (pd\_0 &amp;&amp; ad);

}

return ad;

}, null, null)), (\_l()(), \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_28" /\* ɵted \*/](null, ['logoff']))], null, function (\_ck, \_v) {

var \_co = \_v.component;

var currVal\_0 = \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_29" /\* ɵunv \*/](\_v, 0, 0, \_\_WEBPACK\_IMPORTED\_MODULE\_1\_\_angular\_core\_\_["\_25" /\* ɵnov \*/](\_v, 1).transform(\_co.user));

\_ck(\_v, 0, 0, currVal\_0);

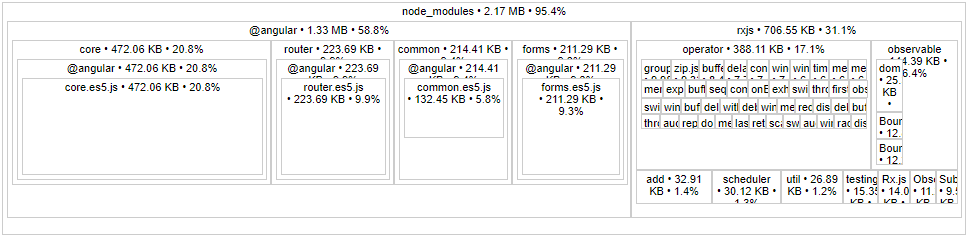
});

}

Sur la même application, la compilation AoT donne les bundles suivants :

* main.bundle.js : 159k (27k en minifié)
* vendor.bundle.js : 2281k (610k en minifié)

On gagne énormément sur la taille du fichier vendor.bundle.js, puisque le compiler Angular n'est plus embarqué, et on a un fichier main.bundle.js sensiblement de la même taille (pour la version minifiée).



Les avantages de cette compilation sont clairs : on gagne en taille de bundle et on gagne en performance (étant donné que la phase de compilation des templates n'a pas à être effectuée). L'autre avantage important est que l'on va être capable de détecter les erreurs de syntaxe dans les templates (appel à une méthode du component inexistante, utilisation d'une variable non déclarée, etc.) lors de la phase de build, puisque les templates vont être compilés, plutôt qu'à l'exécution.

Cependant, le temps de build AoT est plus long qu'en JiT. Dans une configuration de développement, où l'on recompile l'application à chaque modification de fichier, ce delta peut s'avérer très pénible. Il est également important de noter qu'un template compilé est plus gros qu'un template HTML, ce qui induit une taille plus importante du bundle main.bundle.js (très généralement compensé par le gain sur le bundle vendor.bundle.js).

## Quelle compilation je dois privilégier ?

En phase de développement, il est plus commode d'utiliser la compilation JiT. Le temps de build réduit permet d'avoir une meilleure fluidité dans le développement.

Par contre, dès que l'on passe sur un environnement hors développement (recette, pré-prod, prod, etc.), la compilation AoT est indispensable. Plus performante, bundle plus petit, et validation des templates, on gagne à tous les niveaux.

Bonnes compilations.

https://sebastienollivier.fr/blog/angular/angular-jit-vs-aot

===========================

**Angular formation :**





