



Une formation  
**Alphorm**.com

## Angular (v4.0)



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Cursus Angular





Une formation  
**Alphorm**.com

## Plan

### Introduction

- Les Web Components
- Architecture d'une application Angular
- Une première application Angular
- Travail avec les templates
- RxJS
- Les formulaires
- Travail avec HTTP
- Le Routage
- Le projet TodoList

### Conclusion



Une formation  
**Alphorm**.com

## Public concerné

Développeurs Web et Architectes



## Connaissances requises

### Formation JavaScript (1/2): Acquérir les fondamentaux

Maîtrisez les fondements du langage JavaScript : DOM, POO, Design Patterns, Formulaires, Hoisting...



### Formation TypeScript : Le guide complet

Soyez plus productif en javascript, découvrez et maîtrisez le langage TypeScript.



Une formation  
**Alphorm**.com

A vous de jouer !





Une formation  
**Alphorm**.com

## Présentation du projet de la formation



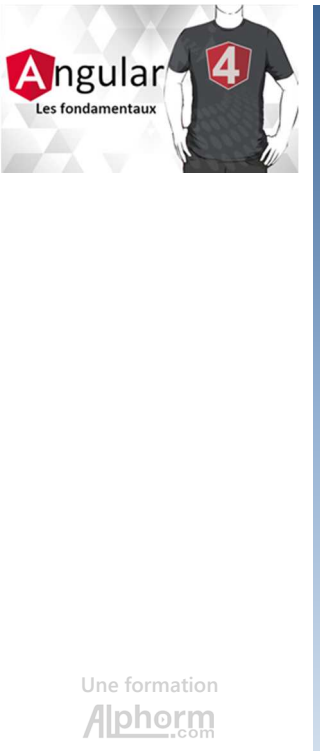
Frédéric GAURAT



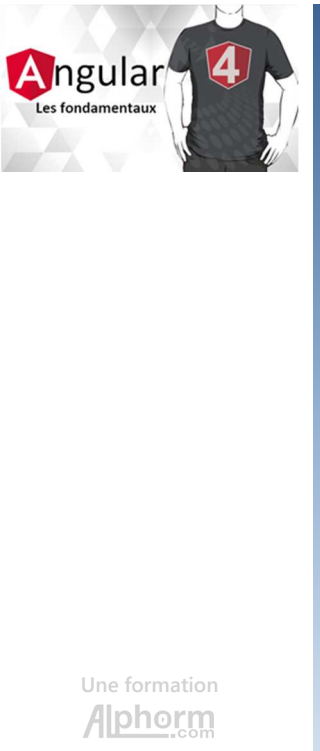
Une formation  
**Alphorm**.com

## Plan

Le projet Todo List  
Le serveur et la base de données  
Présentation des outils



# Le projet Todo List



# Le serveur et la BDD





## Présentation des outils

Microsoft Visual Studio Code  
Ng-cli (<https://cli.angular.io/>)

Une formation  
**Alphorm**.com

# Merci



Une formation  
**Alphorm**.com

# Création de balises personnalisées



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Les Web Components  
Templates  
Shadow DOM  
Custom elements



## Les Web Components



Ensembles de spécifications ([W3C](#)) définissant des API permettant la création de balises réutilisables

```
<alphorm>...</alphorm>
```

Une formation  
**Alphorm**.com



## Templates

La spécification des Templates, décrit la création de fragment HTML

Ces fragments ne sont pas chargés au démarrage de la page

Ces fragments sont instanciés plus tard

Une formation  
**Alphorm**.com





Une formation  
**Alphorm**.com

## Shadow DOM

La spécification **Shadow DOM** décrit comment encapsuler les éléments ou les styles de vos composants (rendre privé)



Une formation  
**Alphorm**.com

## Custom elements

La spécification **Custom elements** définit comment concevoir et utiliser des nouveaux types d'élément DOM

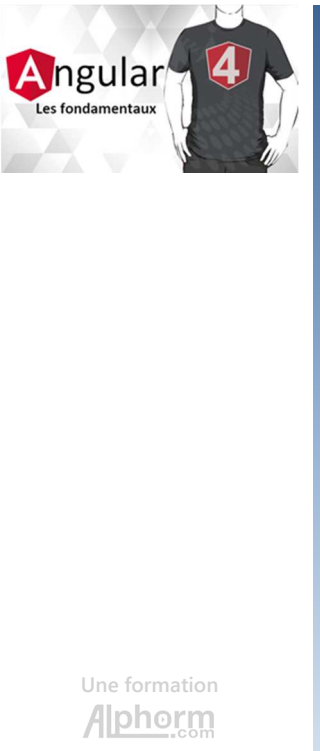


Une formation  
**Alphorm**.com

## Exemple HelloWorld



# Merci

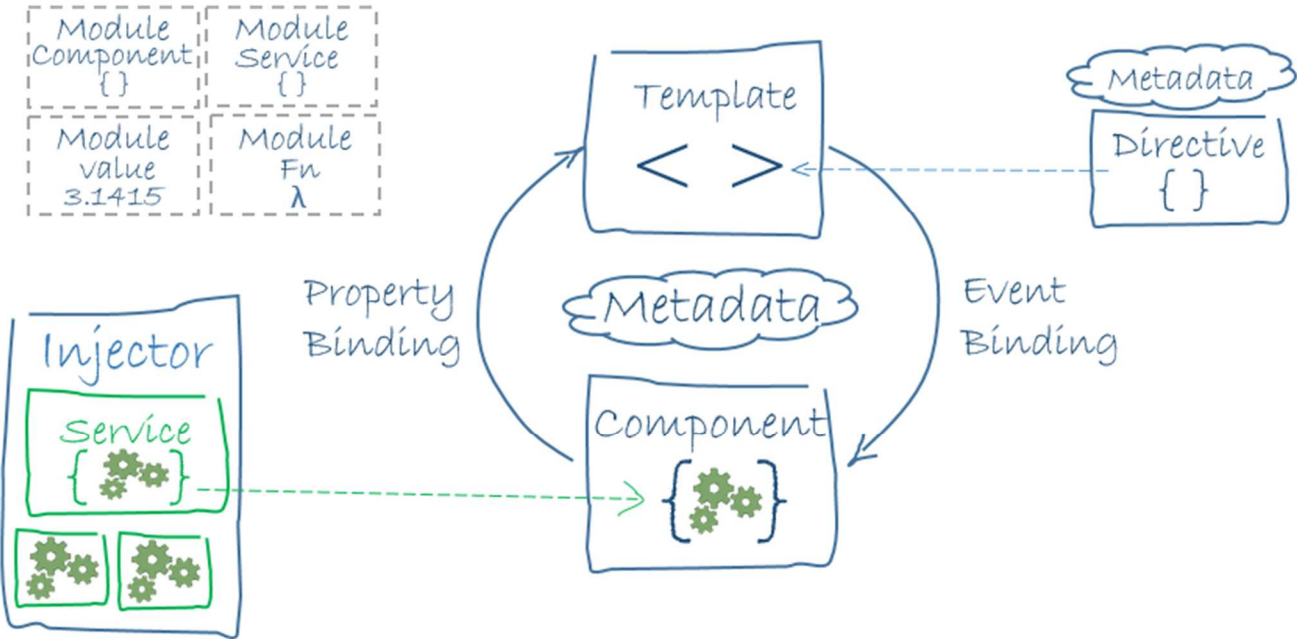


# Architecture d'une application Angular



Frédéric GAURAT

Une formation  
**Alphorm**.com



# Merci



Une formation  
**Alphorm**.com

## Les Modules



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Organiser une application avec des modules

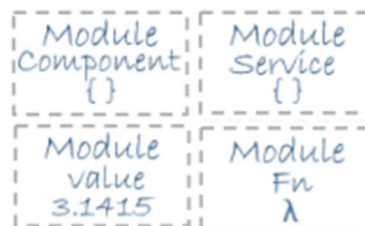
Le module root

Structure d'un module



Une formation  
**Alphorm**.com

## Organiser via des modules



Un module définit la façon dont les  
éléments d'une application s'intègrent  
C'est un top-level component



## Le module root

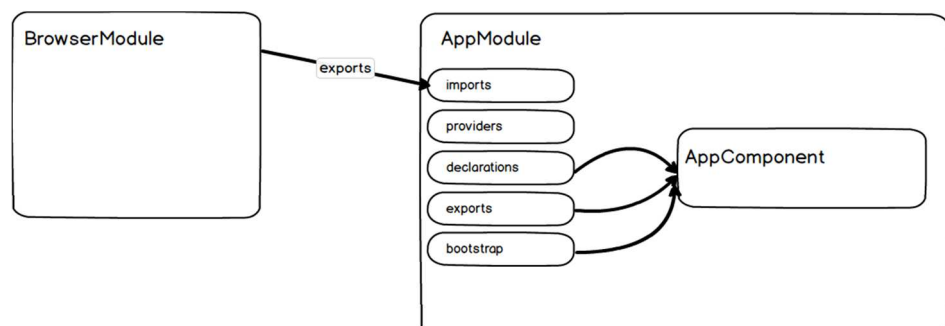
Une application a au moins un module **principal**

Ce module est utilisé pour **initialiser** l'application

Une formation  
Alphorm.com



## Structure d'un module



Une formation  
Alphorm.com

# Merci



Une formation  
**Alphorm**.com

## Les Components



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Les Components  
Les Templates  
Metadata  
Le Data binding  
Conclusion



Une formation  
**Alphorm**.com

## Les Components

Un Component contrôle un fragment de  
l'interface utilisateur  
Il est défini dans une classe  
La classe interagit avec la view via ses  
propriétés et ses méthodes





## Les Templates

Les templates définissent la partie visible de votre component

Un template ressemble à un fichier HTML avec quelques différences : la template syntax

Une formation  
**Alphorm**.com



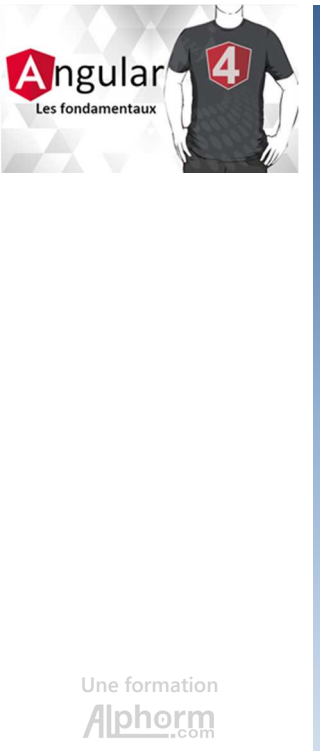
## Metadata

Une simple classe ne suffit pas pour décrire un component

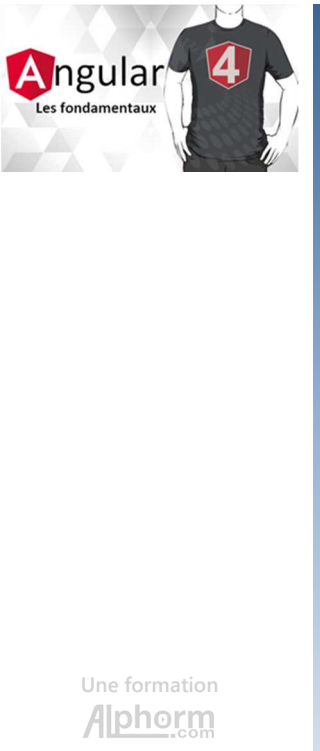
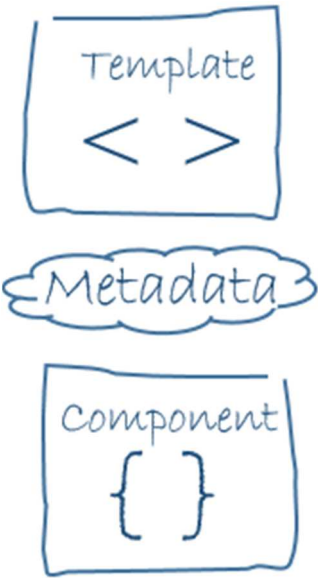
Il faut compléter la classe avec un **decorator** (TypeScript)

Les metadatas sont un lien entre component et template

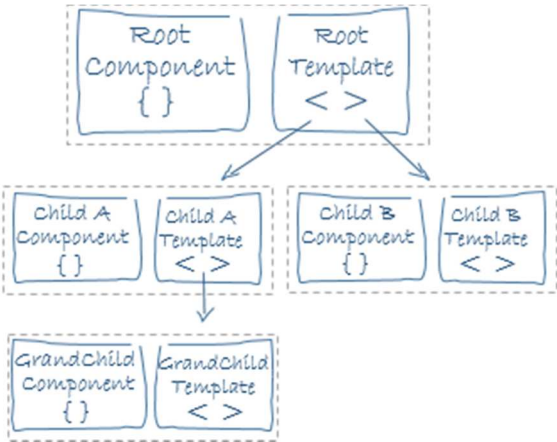
Une formation  
**Alphorm**.com

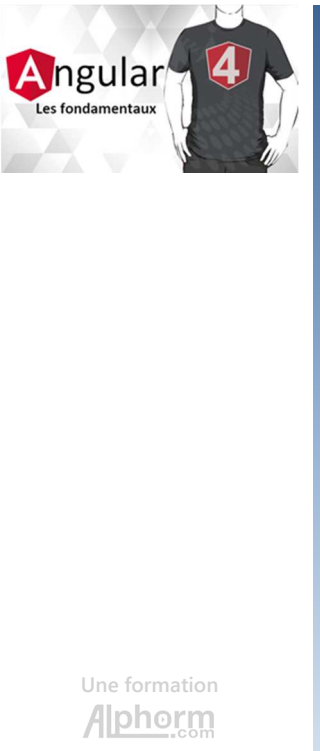


# Metadata

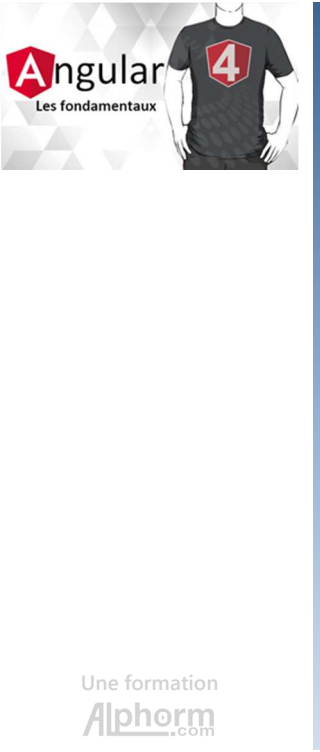
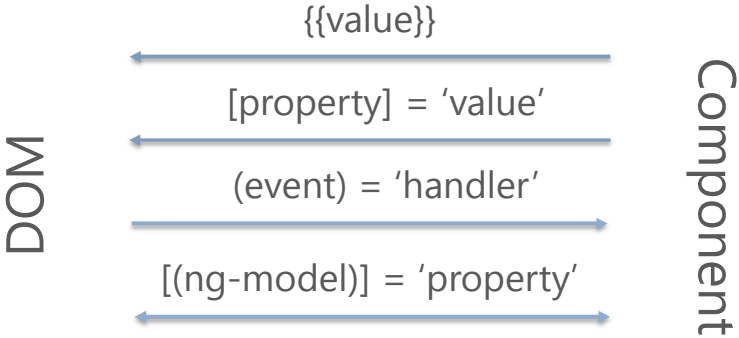


# Relations entre Components

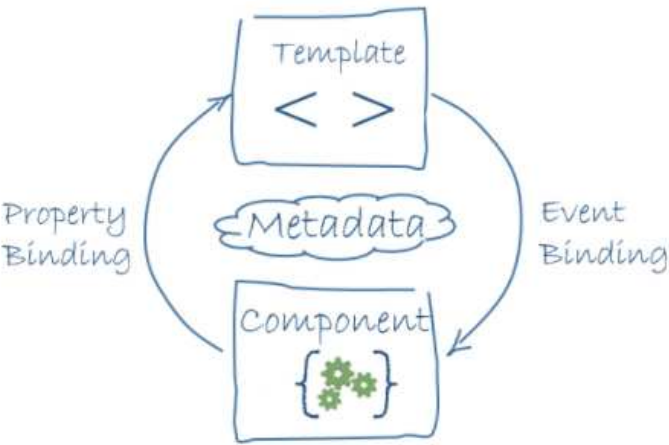




# Le Data binding



# Conclusion



# Merci



Une formation  
**Alphorm**.com

## Les Directives



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Introduction

Les 2 types de Directives

Structural directives

Attribute directives



Une formation  
**Alphorm**.com

## Introduction

Les directives s'utilisent dans les templates

Les components sont des directives (AngularJS) mais sont considérés comme des éléments à parts entières

Nous utilisons des directives pour structurer nos templates et pour changer le comportement d'un élément



## Les 2 types de Directives

Structural directives  
Attribute directives

Une formation  
**Alphorm**.com



## Structural directives

Les directives structurelles  
permettent de modifier le layout en  
manipulant les éléments du DOM

```
<li *ngFor="let hero of heroes"></li>  
<hero-detail *ngIf="selectedHero"></hero-detail>
```

Une formation  
**Alphorm**.com



## Attribute directives

Les directives attribut sont utilisées pour modifier l'apparence ou le comportement d'un élément

```
<input [(ngModel)]="hero.name">
```

Une formation  
**Alphorm**.com

# Merci



Une formation  
**Alphorm**.com

## Les Services



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Les Services

La notion de service regroupe valeurs, fonctions ou fonctionnalités nécessaires à votre application

En règle générale, c'est une simple classe implémentant une fonctionnalité





## Les Services

Un service sera injecté dans un component  
Il aura la responsabilité de décharger le  
component de tout ce qui ne concerne pas  
l'interface utilisateur

Une formation  
**Alphorm**.com

**Merci**



Une formation  
**Alphorm**.com

# Notion d'injection de dépendance



Frédéric GAURAT



Une formation  
**Alphorm**.com

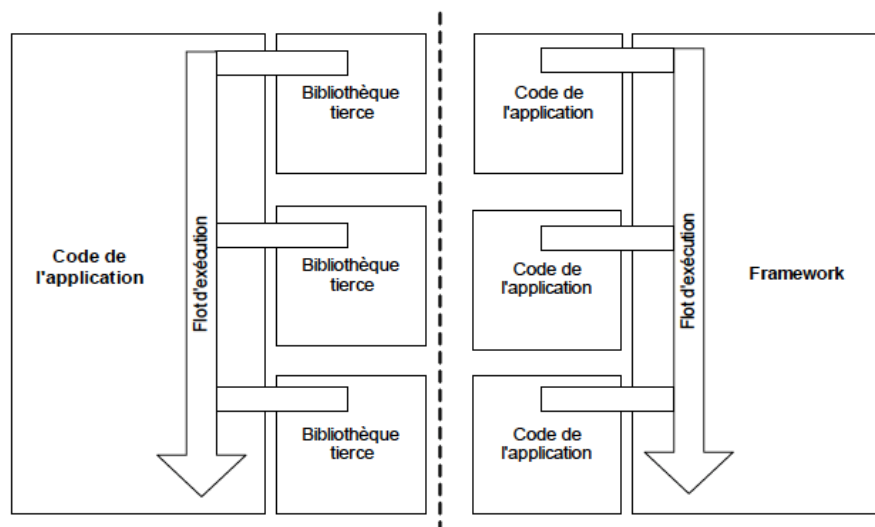
## Plan

L'inversion de contrôle  
L'injection de dépendance  
L'injector



Une formation  
**Alphorm**.com

## L'inversion de contrôle



Une formation  
**Alphorm**.com

## L'injection de dépendance

L'injection de dépendance est le moyen de fournir à un component un objet construit avec toutes ses dépendances

```
Component service  
{Constructor(service)}
```



## L'injector

L'injector est sollicité par Angular lors de la création d'un component

Il maintient un conteneur d'instance précédemment créée

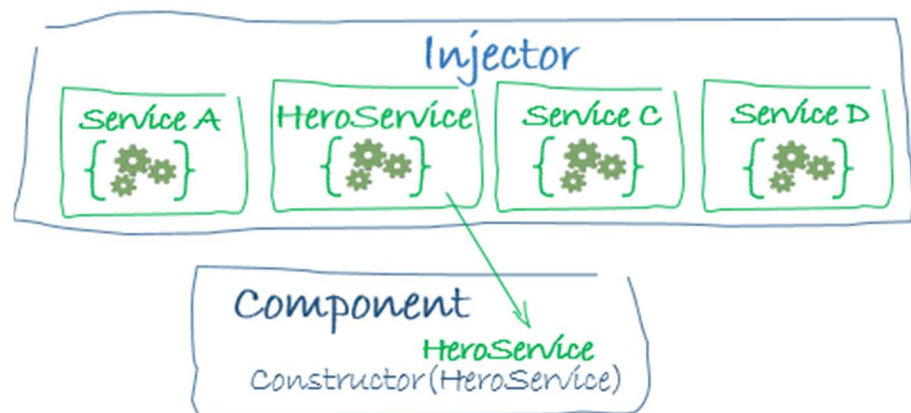
Si le service demandé n'existe pas, il est alors instancié et passé en paramètre au constructeur du component

Sinon il est directement passé en paramètre au constructeur du component

Une formation  
**Alphorm**.com



## L'injector



Une formation  
**Alphorm**.com

# Merci



Une formation  
**Alphorm**.com

## Création du projet



Frédéric GAURAT



## Plan

HelloWorld !  
QuickStart seed  
Test de l'application

Une formation  
**Alphorm**.com



## HelloWorld !



Une formation  
**Alphorm**.com



Une formation  
**Alphorm**.com

## QuickStart seed

Installation de l'environnement

Téléchargement des fichiers

**npm install**

Les fichiers essentiels

**app/app.component.ts**

**app/app.module.ts**

**main.ts**

**package.json**



Une formation  
**Alphorm**.com

## Test de l'application

Lancement de l'environnement de test

**npm run start**

Présentation de **browser-sync**

# Merci



Une formation  
**Alphorm**.com

## Démarrer “from scratch” avec angular-cli



Frédéric GAURAT





Une formation  
**Alphorm**.com

## Plan

Création d'un projet avec angular-cli  
Générer un composant  
Tester une application localement  
L'outil webpack  
Les autres outils



Une formation  
**Alphorm**.com

## Projet avec angular-cli

<https://cli.angular.io/>

```
> npm install -g @angular/cli  
> ng new my-dream-app  
> cd my-dream-app  
> ng serve
```



## Générer un composant

La commande **ng generate**

Une formation  
**Alphorm**.com



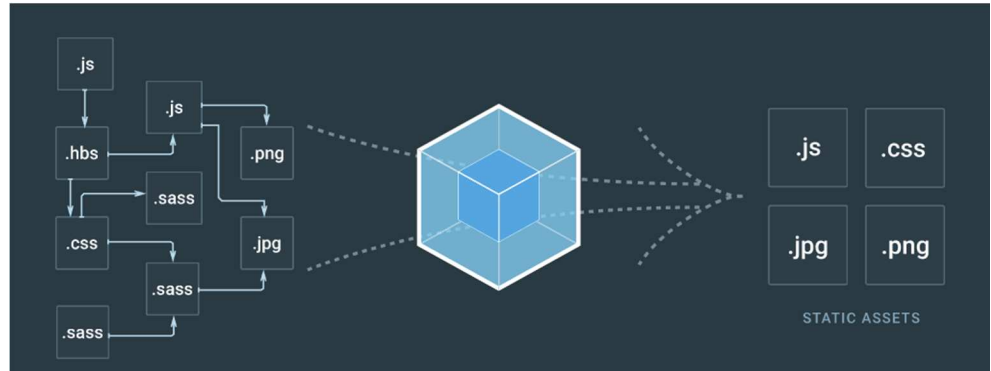
## Tester une app localement

La commande **ng serve**

Une formation  
**Alphorm**.com



## L'outils webpack



Une formation  
**Alphorm**.com



## Les autres outils

Lancement des tests unitaires  
Vérification du code avec **lint**

Une formation  
**Alphorm**.com

# Merci



Une formation  
**Alphorm**.com

## Notion d'interpolation



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Interpolation

Template expressions



Une formation  
**Alphorm**.com

## Interpolation

Une interpolation permet l'évaluation de variables ou d'expressions à l'intérieur d'une chaîne de caractères littérale

Avec Angular l'interpolation se place entre accolades  
Le texte entre accolade est une **template expression**

Exemples :

La somme de 1 + 1 est {{1 + 1}}

Hello {{name}}



Une formation  
**Alphorm**.com

## Interprétation d'une Interpolation

Evaluation du contenu

conversion du résultat en une  
chaîne de caractères

Assignation du résultat par  
**property binding** (modification du  
DOM)

**Merci**



Une formation  
**Alphorm**.com

## Utilisation des bindings



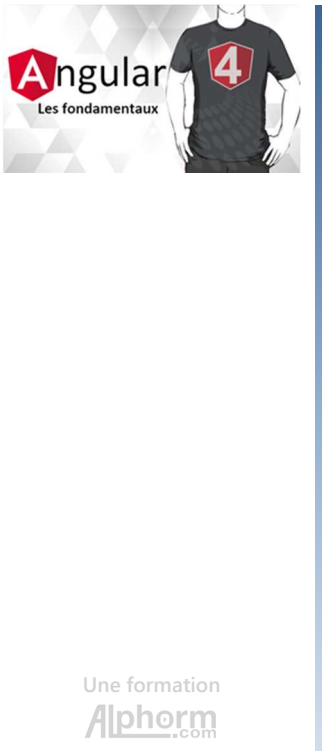
Frédéric GAURAT



Une formation  
**Alphorm**.com

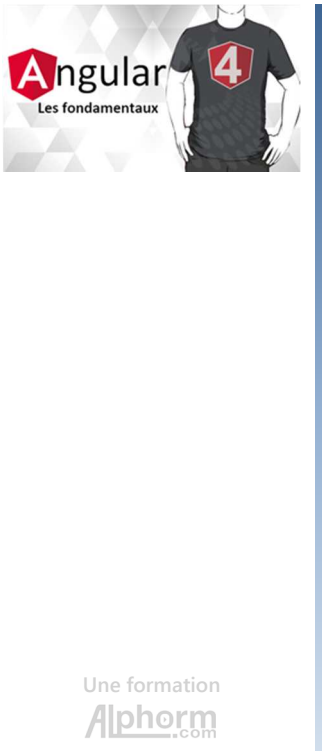
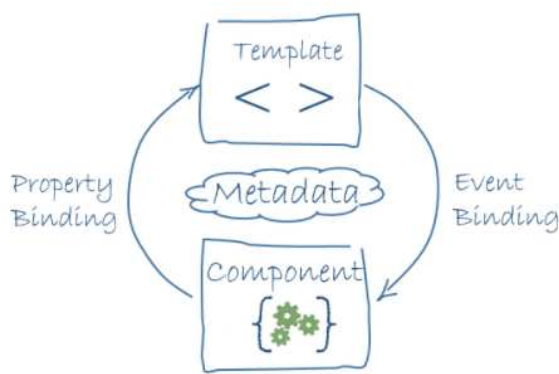
## Plan

Le principe de binding  
Les différents types de binding  
Attribut HTML et propriété DOM  
Les cibles du binding



# Le principe de binding

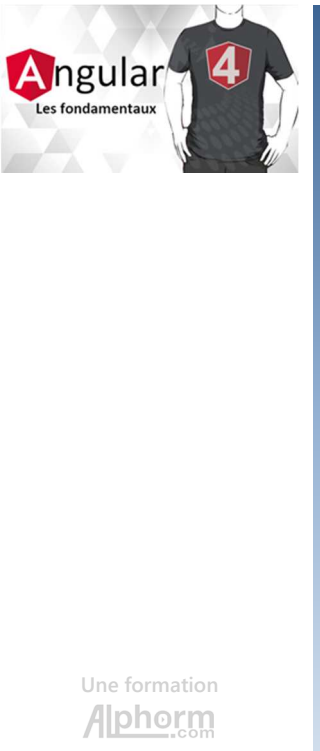
Direction du binding



# Les types de binding

Data direction	Syntax	Type
One-way from data source to view target	<code>{{expression}}</code> <code>[target]="expression"</code> <code>bind-target="expression"</code>	Interpolation Property Attribute Class Style
One-way from view target to data source	<code>(target)="statement"</code> <code>on-target="statement"</code>	Event
Two-way	<code>[(target))]="expression"</code> <code>bindon-target="expression"</code>	Two-way



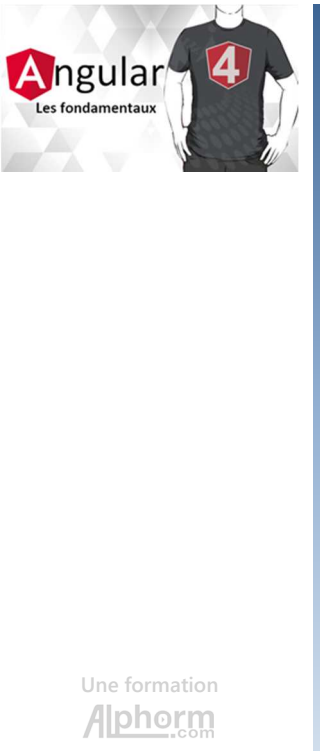


## Attribut HTML et propriété DOM

Un attribut HTML est différent d'une propriété DOM !

Un attribut HTML initialise une propriété DOM

Une propriété DOM peut varier, pas un attribut HTML



## Les cibles du binding

Type	Target	Examples
Property	Element property Component property Directive property	<pre>&lt;img [src]="heroImageUrl"&gt; &lt;hero-detail [hero]="currentHero"&gt;&lt;/hero-detail&gt; &lt;div [ngClass]="{special: isSpecial}"&gt;&lt;/div&gt;</pre>
Event	Element event Component event Directive event	<pre>&lt;button (click)="onSave()"&gt;Save&lt;/button&gt; &lt;hero-detail (deleteRequest)="deleteHero()"&gt;&lt;/hero-detail&gt; &lt;div (myClick)="clicked\$event" clickable&gt;click me&lt;/div&gt;</pre>
Two-way	Event and property	<pre>&lt;input [(ngModel)]="name"&gt;</pre>
Attribute	Attribute (the exception)	<pre>&lt;button [attr.aria-label]="help"&gt;help&lt;/button&gt;</pre>
Class	class property	<pre>&lt;div [class.special]="isSpecial"&gt;Special&lt;/div&gt;</pre>
Style	style property	<pre>&lt;button [style.color]="isSpecial ? 'red' : 'green'"&gt;</pre>

# Merci



Une formation  
**Alphorm**.com

## Directives, Pipe et template variable



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Notion de directives structurelles  
Quelques directives Angular  
L'opérateur Pipe  
Les variables de template



Une formation  
**Alphorm**.com

## Directives structurelles

Elles sont responsable de la structure du DOM  
Elles s'appuient sur la spécification  
Templates du W3C  
Pour en faciliter l'utilisation on utilise une  
microsyntax préfixé par un '\*'



Une formation  
**Alphorm**.com

## Quelques directives Angular

**\*ngIf** : ajout ou suppression conditionnel d'élément DOM

**\*ngFor** : permet de répéter un template en s'appuyant sur une liste

**\*ngSwitch** : permet d'afficher différentes vues en fonction d'une condition



Une formation  
**Alphorm**.com

## L'opérateur Pipe

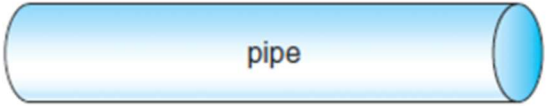
Reprend l'idée du pipe Unix : ( symbole | )

La sortie d'un processus devient l'entrée du suivant

Permet la transformation du résultat d'une expression



## L'opérateur Pipe

hello  HELLO

uppercase

{{« hello »|uppercase}}

Une formation  
**Alphorm**.com



## Les variables de template

Les variables de template sont des références à des éléments dans le DOM (html, component ou directives)  
Elles sont préfixées par un '#'

Une formation  
**Alphorm**.com

# Merci



## La programmation réactive avec RxJS



Frédéric GAURAT

Une formation  
**Alphorm**.com



Une formation  
**Alphorm**.com

## Plan

Qu'est-ce que la programmation réactive?  
Le pattern Observer  
Le pattern Iterator  
Introduction à RxJS



Une formation  
**Alphorm**.com

## La programmation réactive

Un modèle de programmation visant à conserver une cohérence d'ensemble en propageant les modifications d'une source réactive (modification d'une variable, entrée utilisateur, etc.) aux éléments dépendants de cette source

*Source : wikipedia*



# La programmation réactive

Exemple avec Excel

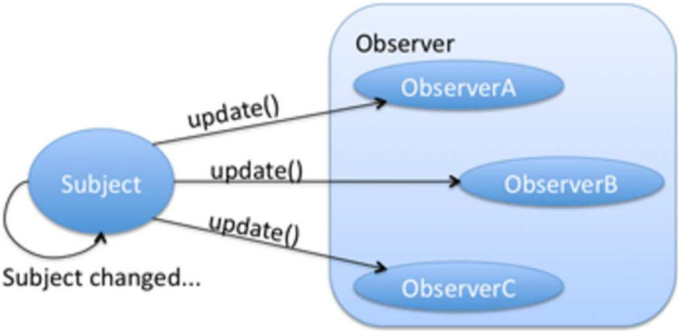
$f_x$  | =C1+D1

C	D	E
2	3	5

Une formation  
Alphorm.com



# Le pattern Observer



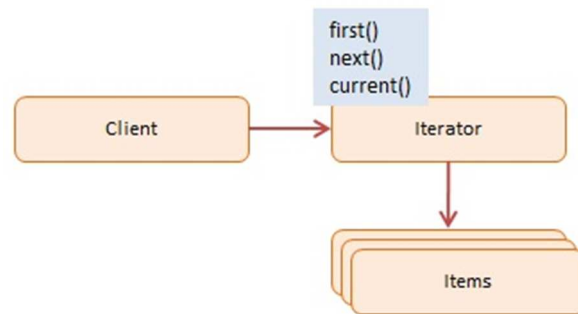
Une formation  
Alphorm.com





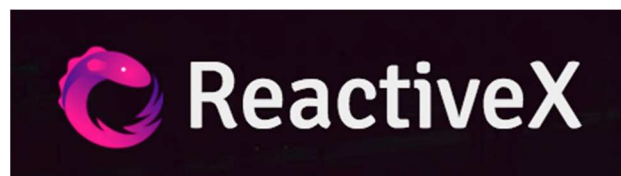
Une formation  
**Alphorm**.com

## Le pattern Iterator



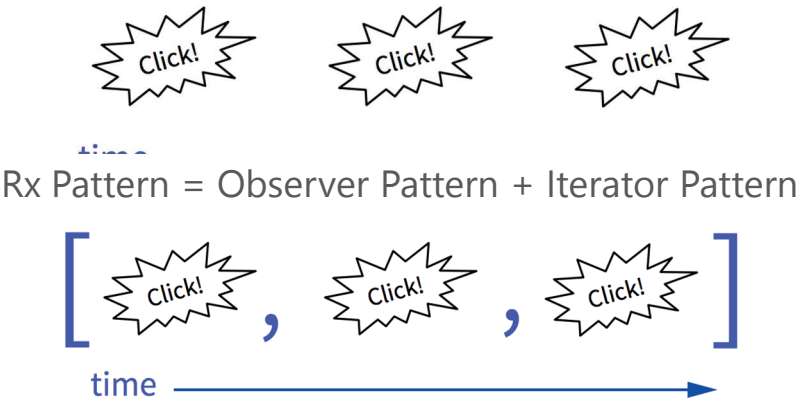
Une formation  
**Alphorm**.com

## Introduction à RxJS





# Introduction à RxJS



Une formation  
**Alphorm**.com

Merci



Une formation  
**Alphorm**.com

# Utilisation de RxJS



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Observable  
Operators



Une formation  
**Alphorm**.com

## Observable

Observable sur un tableau

Observable sur un Event



Une formation  
**Alphorm**.com

## Operators

Un operator est une fonction qui crée un nouvel observer basé sur l'observer de base en y appliquant la fonction

[rxmarbles.com](http://rxmarbles.com)



## Operators

Operator map()

Operator filter()

Operator merge()

Une formation  
**Alphorm**.com

# Merci



Une formation  
**Alphorm**.com

## Construction et validation d'un formulaire



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

- Construire un formulaire
- La directive ngModel
- Surveiller les changements
- Styles CSS et état de formulaire
- Le directive ngSubmit



## Construire un formulaire

Les 2 types de constructions de formulaires

- Template based
- Code based

La class Model

Le form component

Une formation  
**Alphorm**.com



## La directive ngModel

Le template du formulaire

Le binding bi-directionnel (ngModel)

La directive ngForm

Une formation  
**Alphorm**.com



Une formation  
**Alphorm**.com

## Surveiller les changements

ngModel surveille les changements et y affecte des classes

Le champ a été visité : ng-touched (ng-untouched)

La valeur du champ a changé : ng-dirty (ng-pristine)

La valeur du champ est valide : ng-valid (ng-invalid)



Une formation  
**Alphorm**.com

## Styles CSS et état de formulaire

Appliquer des styles CSS aux états renvoyés par ngModel





## Le directive ngSubmit

Exécuter du code lors de la soumission du formulaire

Empêcher la soumission si le formulaire n'est pas valide

Une formation  
**Alphorm**.com

**Merci**



Une formation  
**Alphorm**.com

# Ajax et Angular



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Plan

Utiliser le HttpModule  
Communiquer avec le serveur



Une formation  
**Alphorm**.com

## Le provider Http

Le module doit importer le HttpClientModule

Le service Http est utilisable par injection de dépendance

```
class Http {  
  constructor(_backend: ConnectionBackend, _defaultOptions: RequestOptions)  
  request(url: string|Request, options?: RequestOptionsArgs) : Observable<Response>  
  get(url: string, options?: RequestOptionsArgs) : Observable<Response>  
  post(url: string, body: any, options?: RequestOptionsArgs) : Observable<Response>  
  put(url: string, body: any, options?: RequestOptionsArgs) : Observable<Response>  
  delete(url: string, options?: RequestOptionsArgs) : Observable<Response>  
  patch(url: string, body: any, options?: RequestOptionsArgs) : Observable<Response>  
  head(url: string, options?: RequestOptionsArgs) : Observable<Response>  
  options(url: string, options?: RequestOptionsArgs) : Observable<Response>  
}
```



Une formation  
**Alphorm**.com

## Communiquer avec le serveur

Simuler une web API

Recevoir des données

Envoyer des données

Quelques opérateurs RxJS

# Merci



## Principe de routage, configuration et directive



Frédéric GAURAT

Une formation  
**Alphorm**.com



Une formation  
**Alphorm**.com

## Plan

Fonctionnement du router  
Les éléments de routage  
Configuration



Une formation  
**Alphorm**.com

## Fonctionnement du router

Le router Angular reprend le principe de base du navigateur  
Il interprète une URL pour naviguer vers la bonne vue  
Il peut interpréter les liens pour naviguer vers la bonne vue  
Il log l'historique de navigation pour assurer le fonctionnement des boutons back et forward



## Les éléments de routage

Router outlet

Router links

Router state

Une formation  
**Alphorm**.com



## Router outlet

Endroit réservé où Angular doit placer la vue

```
<router-outlet></router-outlet>  
<!-- Routed views go here -->
```

Une formation  
**Alphorm**.com



Une formation  
**Alphorm**.com

## Router links

**Directive routerLink** : Permet de référencer une route dans le template

**Directive routerLinkActive** : Permet d'ajouter automatiquement une classe si la route est active

```
template: `
  <h1>Angular Router</h1>
  <nav>
    <a routerLink="/crisis-center" routerLinkActive="active">Crisis Center</a>
    <a routerLink="/heroes" routerLinkActive="active">Heroes</a>
  </nav>
  <router-outlet></router-outlet>
`
```



Une formation  
**Alphorm**.com

## Router state

Angular maintient un arbre des routes activées

Cet Arbre est le **RouterState**

Chaque route de cet arbre est accessible et propose des méthodes de parcours



## Configuration

Concrètement, le router doit avoir au moins une route configurée

Les routes sont stockées dans un tableau de Route

Ce tableau est ensuite passé au module RouterModule :

- **RouterModule.forRoot()** (pour des routes principales)
- **RouterModule.forChild()** (pour des sous-routes)

Une formation  
**Alphorm**.com

# Merci





Une formation  
**Alphorm**.com

## Le projet TodoList



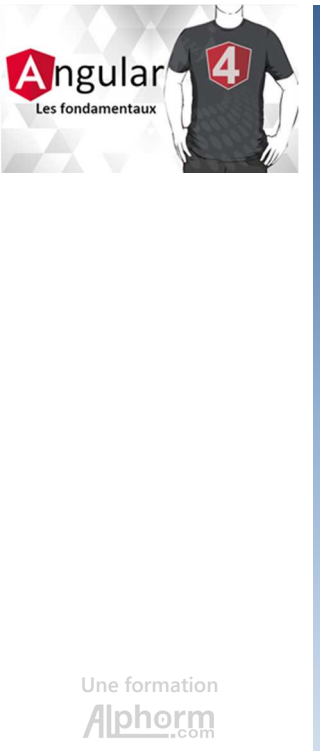
Frédéric GAURAT



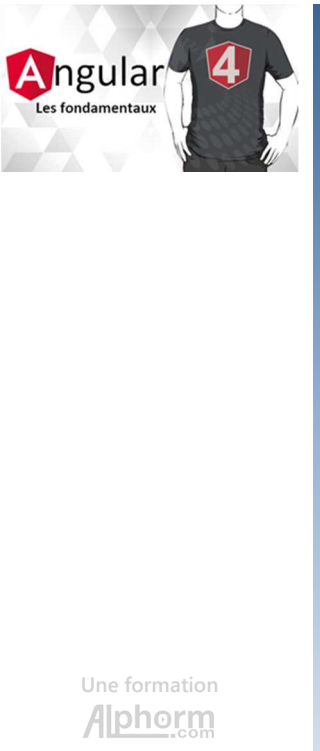
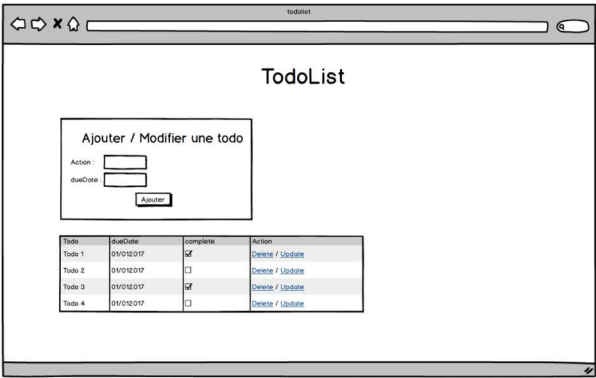
Une formation  
**Alphorm**.com

## Plan

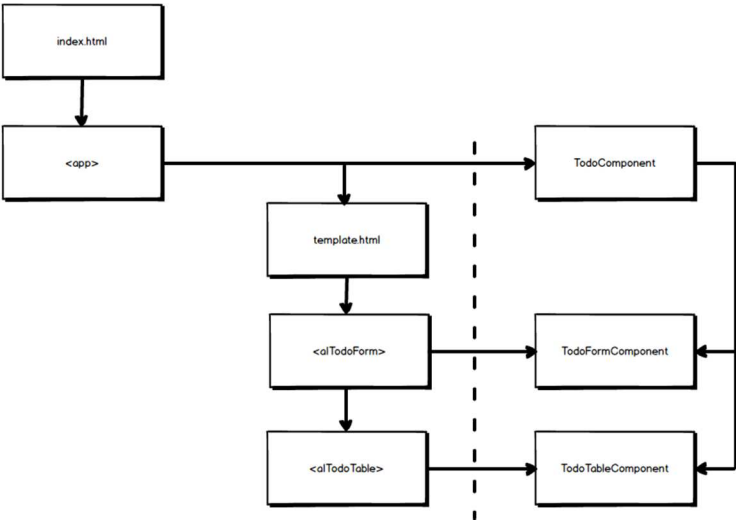
Le projet Todo List  
L'approche orientée composant  
Développement



# Le projet Todo List



# L'approche orientée composant





## Démonstration

Création du projet avec **angular-cli**  
Configuration de la base de données  
(deployd)  
Création des composants

Une formation  
**Alphorm**.com

# Merci



Une formation  
**Alphorm**.com

## Conclusion



Frédéric GAURAT



Une formation  
**Alphorm**.com

## Bilan

Les Web Components  
Architecture d'une application Angular  
Travail avec les templates  
Programmation Reactive avec RxJS  
Les formulaires  
Travail avec HTTP  
Le Routage  
Le Projet TodoList



## Prochaine formation

Angular : Avancé



Une formation  
**Alphorm**.com

# Merci