



# Formation Angular 5

par Rémi Bouguermouh - [@Remify](#)

sources et documentation : <https://github.com/Remify/formation-angular>



# Session 1 - Programme

- Introduction à Angular
- Approche par Composant
- Mise en place et installation
- TP1 Composants



# Qu'est ce que Angular ?

- Framework applicatif Front End (HTML / CSS / JS)
- Cross Platform (Navigateurs et Appareils)
- Plateforme de développement (génération de code, test, CLI, animations, accessibilité)



# Framework applicatif Front End

- **Framework** : Cadre couvrant un large éventail de technologies et de besoins
- **Application** : Toute la puissance du javascript / typescript pour la création d'application navigateurs et desktop
- **Front End** : Au service de l'UI et de l'UX (Manipulation DOM, création de composants, Ajax, modules, routing, formulaires ...)

Package complet avec des approches et des règles. (Opinionated)



# Pourquoi le Javascript ?

- On a que ça
- Natif sur tous les navigateurs
- Vrai langage pour la programmation logiciel suite aux travaux autour du standard ECMAScript et des avancements autour des compilateurs



Les histoires du  
**père Castor**



# Typescript

- Superset du Javascript (on écrit du Javascript)

Ajouts syntaxiques :

- Typage Fort (définition, Inférence de type ...)
- Syntaxe simplifiée par rapport au Javascript (classes, interface, fonctions, enums ...)
- Support features ESNext

**Objectif : Simplifier le développeur d'applications à grandes échelles**



# Les approches Angular

- Déclarative
- Modulaire
- Reactive

->Une approche très Fonctionnelle. (Nouveaux paradigmes)





# Component Driven et Approche déclarative

1 chose à retenir : Qu'est ce qu'un Composant et à quoi ça sert ?

- Extension du vocabulaire HTML
- Élément UI (visuel)
- Réutilisable (modulaire)
- Sans effet de bord (déclaratif)

# Composition d'un composant

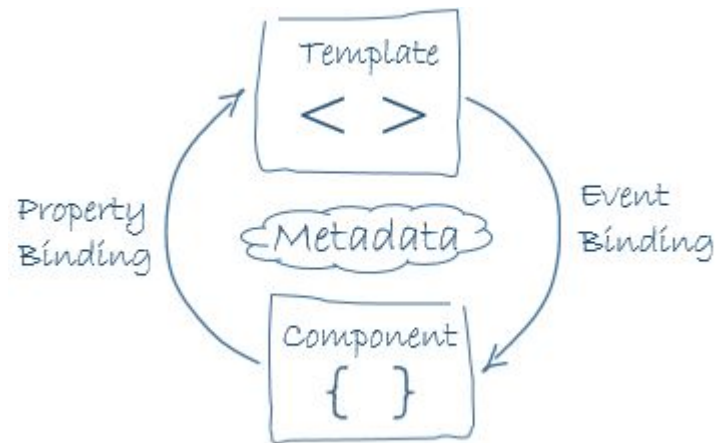
2 Parties :

- Template
- Composant (logique)

2 way binding :

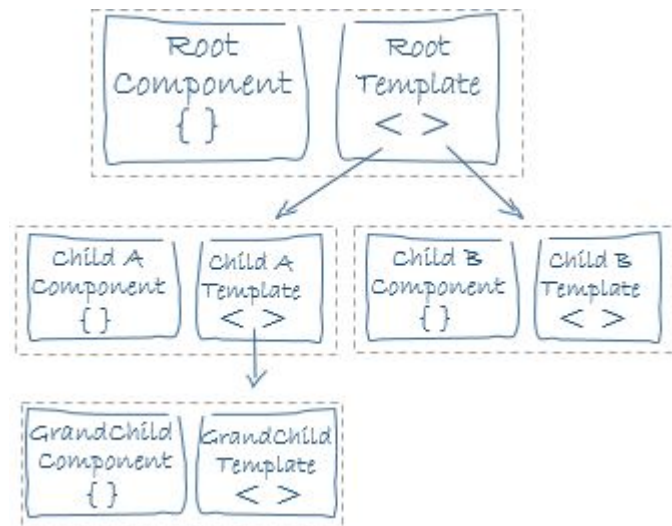
- Passage de propriété (values)
- Événements

<https://angular.io/guide/architecture>



# Arborescence

- Les bindings peuvent être passés à d'autres composants
- Un composant peut en appeler un autre (système de balises)





# Syntaxe

Template : [exemple](#)

Composant : [exemple](#)

Appeler un composant :

```
<app-todo [task]="todo" (onDelete)="onDelete($event)"></app-todo>
```



# Les modules

- Utilisé pour organiser le projet en ensembles autonomes et/ou réutilisable
- Package de composants, services, classes, directives, pipes ...
- Comporte comme un namespace
- Fonctionnalités d'import et d'export pour la communication avec d'autres modules



# Les modules - la syntaxe

- Classe marqué par un Décorateur NgModule

```
@NgModule({  
  declarations: [...],  
  providers: [...],  
  imports: [...],  
  exports: [...]  
})
```



## Les modules - astuces

- Un module par “page”
- Anticiper les problématiques d'accès (SharedModule)
- Attention aux dépendances en boucles
- Séparer les services de données des services logiques.



# Les pipes

- Comporte comme une fonction map
- Chainable
- Passage de paramètre





# Les services

Classe qui joue le rôle d'intermédiaire entre les composants

- Pour les données (échange, http)
- Pour la logique
- Interface avec des librairies / systèmes ...