Formation Angular (2+)

Romain Bohdanowicz
Twitter : @bioub
http://formation.tech/

formation.tech

Introduction

Présentations

- ## Romain Bohdanowicz
  Ingénieur EFREI 2008, spécialité en Ingénierie Logicielle
- ## Expérience
  Formateur/Développeur Freelance depuis 2006
  Plus de 8000 heures de formation animées
- ## Langages
  Expert : HTML / CSS / JavaScript / PHP / Java
  Notions : C / C++ / Objective-C / C# / Python / Bash / Batch
- ## Certifications
  PHP 5 / PHP 5.3 / PHP 5.5 / Zend Framework 1
- ## Particularités
  Premier site web à 12 ans (HTML/JS/PHP), Triathlète à mes heures perdues
- ## Et vous ?
  Langages ? Expérience ? Utilité de cette formation ?

TypeScript

TypeScript - Introduction

- TypeScript : JavaScript + Typage statique
  - TypeScript est un langage créé par Microsoft, construit comme un sur-ensemble d'ECMAScript
  - Pour pouvoir exécuter le code il faut le transformer en JavaScript avec un compilateur
  - A quelques exceptions près et selon la configuration, le JavaScript est valide en TypeScript
  - Le principal intérêt de TypeScript est l'ajout d'un typage statique

TypeScript - Installation

- Installation
  - `npm install -g typescript`
- Création d'un fichier de configuration
  - `tsc --init`
- Compilation
  - `tsc`

Zone.js

Angular CLI

formation.tech

Composants

Composants - Introduction

Composants - Lifecycle Hooks

Composants - Lifecycle Hooks

```
import { Component, OnDestroy, OnInit } from '@angular/core';

@Component({
  selector: 'hello-lifecycle',
  template: `
    {{ now | date:'HH:mm:ss' }}
  `,
})
export class LifecycleComponent implements OnInit, OnDestroy {

  public now = new Date();
  private intervalId: number;

  ngOnInit() {
    this.intervalId = setInterval(() => {
      this.now = new Date();
    }, 1000)
  }

  ngOnDestroy() {
    clearInterval(this.intervalId);
  }

}
```

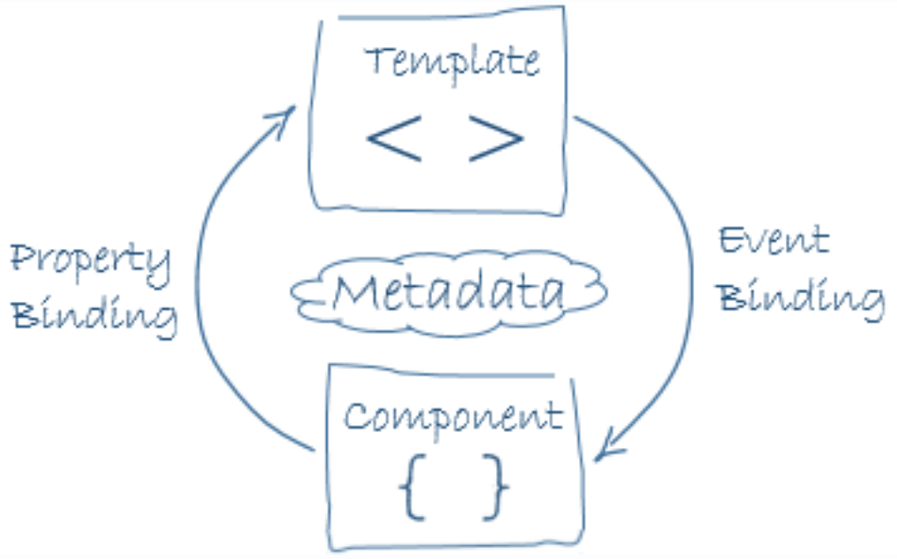
Templates

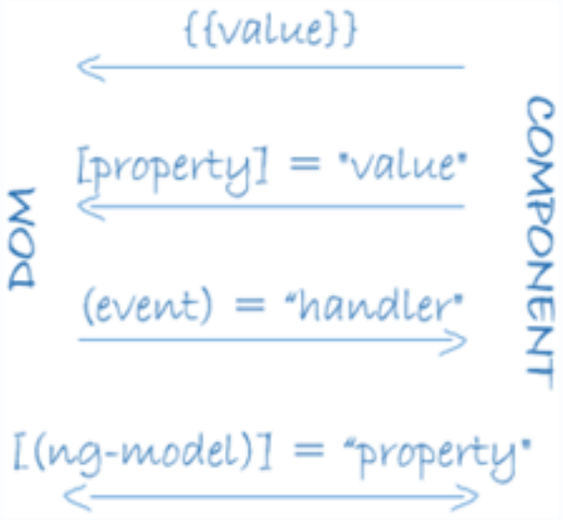
Templates - Introduction

- Templates
  - Comme dans AngularJS, on décrit l'interface de manière déclarative dans des templates
  - Chaque template est compilé par le compilateur d'Angular, soit en amont (mode AOT pour Ahead Of Time Compilation), soit dans le browser (mode JIT pour Just In Time Compilation)
  - Les templates sont ainsi transformé en du code optimisé pour la VM/Moteur JavaScript

Templates - Data binding

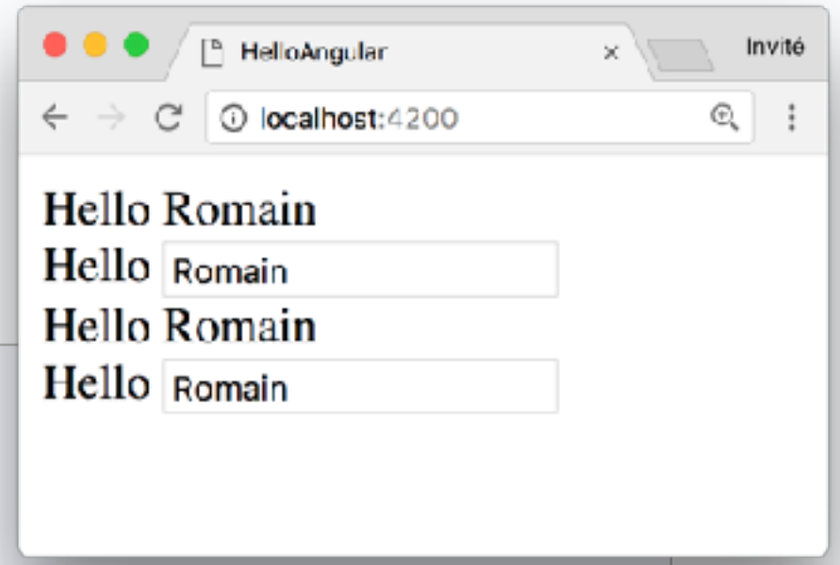
- Data binding
  - Sans data binding ce serait au développeur de maintenir les changements à opérer sur le DOM à chaque événement
  - Dans jQuery par exemple, cliquer sur un bouton peut avoir pour conséquence de rafraîchir une balise, de lancer un indicateur de chargement…
  - Avec Angular le développeur décrit l'état du DOM en fonction de propriétés qui constitue le Modèle, ainsi un événement n'a plus qu'





Templates - Property Binding

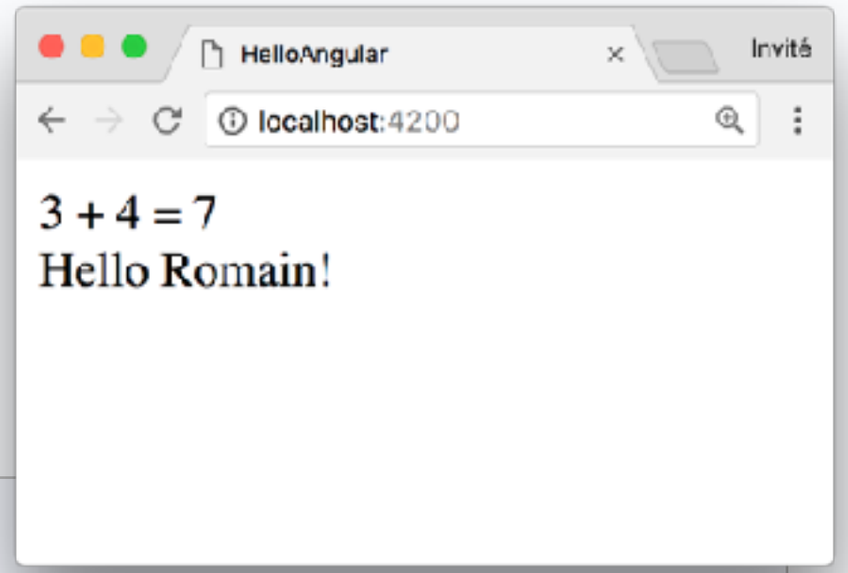
- 2 syntaxes
  Pour synchroniser le DOM avec le modèle (les propriétés publiques du composant dans Angular)
  - bind-nomDeLaPropDuDOM="propDuComposant"
  - [nomDeLaPropDuDOM]="propDuComposant"




```
import { Component } from '@angular/core';

@Component({
    selector: 'hello-property-binding',
    template: `
      <div>Hello <span bind-textContent="prenom"></span></div>
      <div>Hello <input bind-value="prenom"></div>
      <div>Hello <span [textContent]="prenom"></span></div>
      <div>Hello <input [value]="prenom"></div>
    `,
})
export class PropertyBindingComponent {
    public prenom = 'Romain';
}
```

Templates - Expressions

- Dans un property binding il est possible d'utiliser des noms de propriétés ou des expressions, sauf les expressions ayant des effets de bords :
  - affectations (=, +=, -=, ...)
  - new
  - expressions chainées avec ; ou ,
  - incrementation et décrémentation (++ et --)

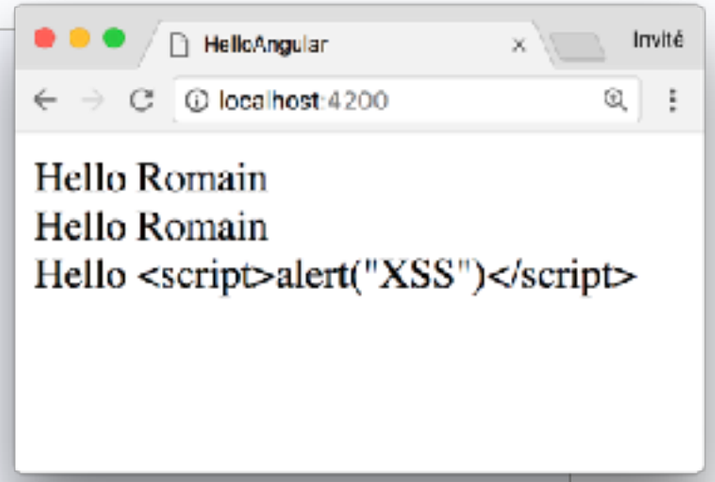



```ts
import { Component } from '@angular/core';

@Component({
    selector: 'hello-prenom',
    template: `
      <div>3 + 4 = <span [textContent]="3 + 4"></span></div>
      <div>Hello <span [textContent]="prenom + '!'"></span></div>
    `,
})
export class PrenomComponent {
    public prenom = 'Romain';
}
```

Templates - Interpolation

- Interpolation
  - Plutôt que bind-innerHTML sur une balise span, on peut utiliser la syntaxe aux doubles accolades {{ }}
  - A privilégier car cette syntaxe échappe les entrées, évitant ainsi que des balises contenues dans les entrées se retrouvent dans le DOM (faille XSS)

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-interpolation',
  template: `
    <div>Hello <span [innerHTML]="prenom"></span></div>
    <div>Hello {{prenom}}</div>
    <div>Hello {{xssAttack}}</div>
  `,
})
export class InterpolationComponent {
  public prenom = 'Romain';
  public xssAttack = '<script>alert("XSS")</script>';
}
```

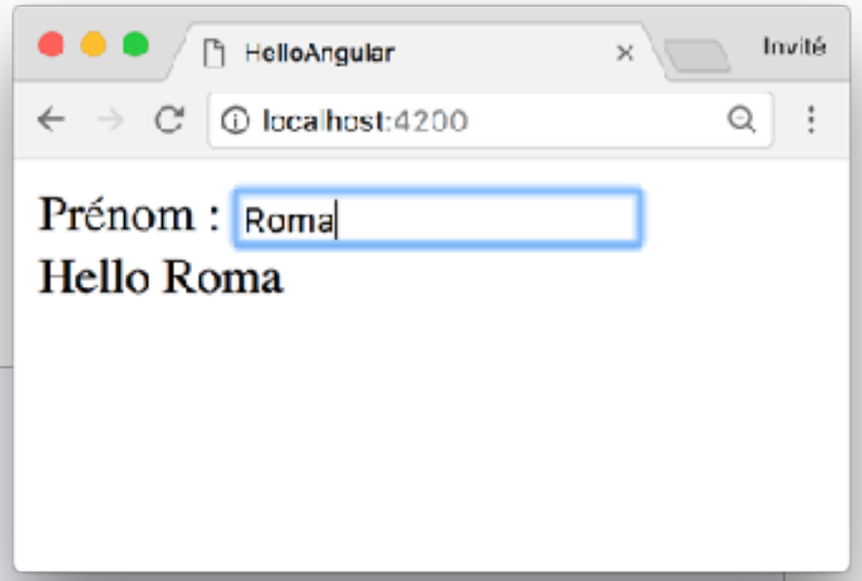


Templates - Event Binding

- 2 syntaxes

  Pour synchroniser le DOM avec le modèle (les propriétés publiques du composant dans Angular), on utilise des événements

  - on-nomEvent="methodeDuComposant()"
  - (nomEvent)="methodeDuComposant()"



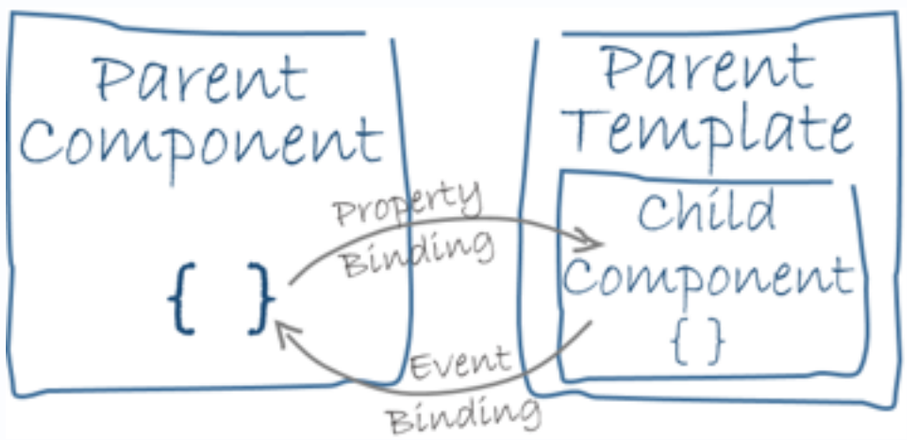

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-event-binding',
  template: `
    <div>Prénom : <input on-input="updatePrenom($event)"></div>
    <div>Prénom : <input (input)="updatePrenom($event)"></div>
    <div>Prénom : <input (input)="prenom = $event.target.value"></div>
    <div>Hello {{prenom}}</div>
  `,
})
export class EventBindingComponent {
  public prenom = '';

  public updatePrenom(e) {
    this.prenom = e.target.value;
  }
}
```

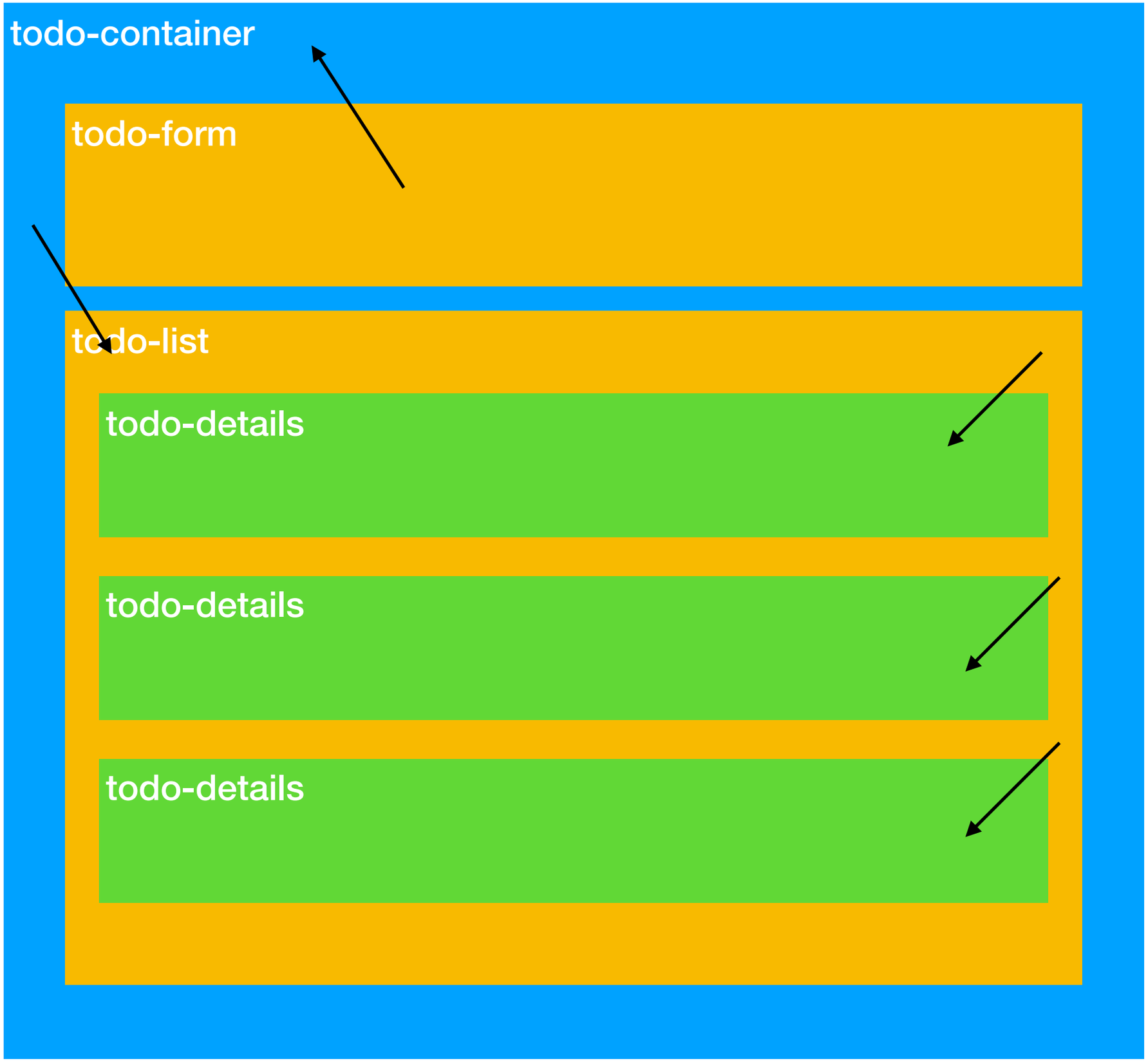
TODO

- Data-binding dans les 2 sens
- TODO
- [( )] = BANANA IN A BOX

TODO

- Communication inter-composant




todo-container
todo-form
todo-list
todo-details
todo-details
todo-details

formation.tech

Détection du changement

Détection du changement - Introduction

- Data-binding
  - Introduit par EmberJS puis repris par les autres frameworks populaire (AngularJS, React, VueJS, Angular)
  - Permet de tenir synchronisé le modèle de données avec la vue (le template), c'est l'implémentation du pattern MVVM (Model / View / ViewModel)
- Détection du changement
  - Pour maintenir la synchronisation il faut un algorithme de détection de changement
  - Dans Angular, à chaque événement, une comparaison est faite entre les propriétés de l'ensemble des composants et leurs valeurs précédentes
  - Si ancienneValeur === nouvelleValeur, le DOM est mis à jour, sinon non

Détection du changement - Introduction

- TODOS slides sur arbres de detection et OnPush

formation.tech

Modules

formation.tech

Pipes

Router

formation.tech

Services

formation.tech

Observables

HttpClient

formation.tech

Tests Automatisés

formation.tech

Best Practices

Best Practices - Linters

Best Practices - Style Guide