

Module Angular

Partie 1

Maxime Tournier

1. Présentation
2. Histoire
3. Démarrage Angular
4. TypeScript
4. Premier pas sur Angular
5. Composants
6. Template

Angular est un **FrameWork**

Traduction

Framework = Cadre de travail

En tant que développeur, on fait souvent la même chose

Exemple:

Valider les formulaires — Gérer la navigation — Traiter les erreurs

Et pour régler ce problème au lieu d'aller récupérer les fonctions d'autres projet on a créé les frameworks

Avantage : Tout le monde travaille sur les mêmes fonctions

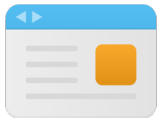
Pour comprendre le fonctionnement d'angular

Il faut comprendre le web aujourd'hui

Site Web / Application Web

Il s'agit bien de deux choses différentes

Avant ça comment ça marche :

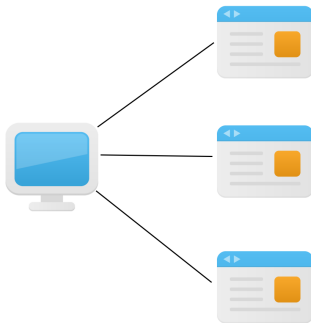


- Fichier client (CSS, HTML, JS)



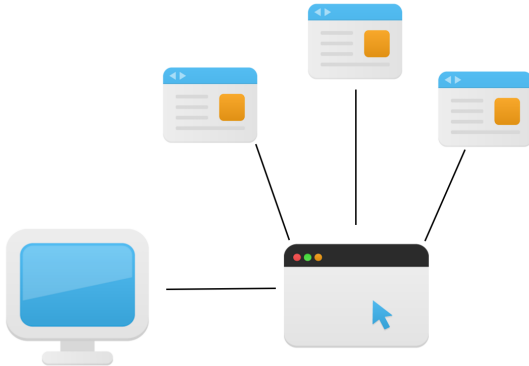
- Fichier serveur (php, java) (ici qu'on fait des requêtes sql)

Et donc un site web fonctionne comme ça :



Présentation

Et une application web marche comme ça:



Resumé :

le client doit faire une requête au serveur a chaque fois dans un site web

alors que dans une application web, nous allons déléguer le travaille à Javascript qui va désactiver ou activer une partie du code html/css

Information

Cette méthode s'appelle SPA (Single Page Application)

AngularJS n'est pas égal à Angular

AngularJS a été créé par Google

AngularJS avait comme architecture MVC

AngularJS n'est plus maintenu depuis 2018

MVC

Module Vue Controlleur (Comme Symfony)

Angular est un framework orienté composant

Nous allons coder une multitude de petits composants qui formera une application

Un composant est une partie du site qui fonctionne de manière autonome sur une application

Angular est-il difficile à apprendre ? :

Angular a une réputation de framework difficile

Car Angular est basé sur Typescript une langue basée sur Javascript qui permet de Typé vos variables

Angular est basé sur la version de Javascript ES6

Regarder la documentation S'IL VOUS PLAÎT
[Angular.io](https://angular.io)

Démarrage Angular

1. Installer un environnement de développement
2. Génère un socle angular (Grâce à Angular-CLI)
3. Nous allons jouer avec le composant racine =)

Information

un composant c'est comme une partie de l'application web. Mais il y a un premier composant à l'origine de tout, que nous appelons le composant racine

Démarrage Angular

Ce qu'on a besoin :

D'abors on va avoir besoin du moteur javascript (NodeJS) et d'un gestionnaire de paquets (NPM)

D'un éditeur de code TypeScript

Un outil qu'on appelle Angular CLI (Qui va nous permet de gérer notre projet Angular)

NodeJS et NPM sont utilisés pour la majorité des projets Javascript moderne

NodeJS permet d'exécuter du code JS côté serveur

NPM

permet de gérer les dépendances de paquets de javascript (angular est un paquet)

NodeJS intègre par défaut NPM

Pour installer NodeJS (et donc aussi NPM)

il vous suffit d'installer sur le site officiel <https://nodejs.org/fr>

Information

Prenez la version LTS

On va avoir besoin d'un IDE

Un IDE est un environnement de développement (Editeur de code amélioré)

- Visual Studio Code : Recommander par Microsoft (Gratuit)
- WebStorm : Beaucoup plus puissant et recommandés par Google, de la suite JetBrains (Payant)

Choisissez celui que vous préférez ou que vous êtes le plus à l'aise

Information

La suite JetBrains (WebStorm) est gratuite pendant tout au long de votre formation, inscription sur leurs sites internet

Pour votre terminal :

Souvent il peut avoir des problèmes avec le terminal Windows

Essayer de toujours d'utiliser PowerShell

Démarrage Angular

Activité :

- Installer NodeJS et NPM sur le site officiel (<https://nodejs.org/fr>)
Prenez la version LTS pour NodeJS
- Installer un IDE : Visual Studio Code ou WebStorm

Pour tester si NodeJS et NPM son bien installer, Ouvrez un terminal

```
$ node -v  
v18.15.0  
$ npm -v  
9.5.0
```

TypeScript est une surcouche de JavaScript

Typescript est développé par Microsoft depuis 2012

TypeScript support les dernier version de JS ES6

TypeScript a besoin d'être compilé (Build) pour fonctionner

Il sera transformé en un fichier JS

L'extension de TS est .ts

Pour compiler Typescript on exécutera cette commande

```
tsc file.ts
```

Cette commande nous créera un fichier JS du nom de file.js

Typescript ajoute la déclaration de variable:

```
myVar: number;  
myVarAssigned: string = "Maxime";  
myClient: IClient = new Client();  
myPersons: Array<Person> = []  
myObject: any;
```

Typescript ajoute la déclaration de variable:

number = Int ou Float

string = Une chaint de caracter

IClient = Un object de type IClient

Array<Person> = Un tableau de "Person"

any = Autre

Activité :

- Installer de quoi compile Typescript avec cette commande

```
npm install -g typescript
```

Activité :

- Refaire l'activité 1 du rappel JavaScript
En typant les variables avec Typescript
- Puis compiler le fichier Typescript et tester le site

Activité :

Mini-Quiz à Choix Multiples

Premier pas avec Angular

Angular-CLI :

Cette application va installer tout ce dont on a besoin
en une seule ligne de commande

et avec cet même outils on va pouvoir piloter certaines fonctionnalités
d'angular directement en ligne de commande

Information

Ce n'est pas la seule méthode pour démarrer un projet Angular, mais bien
la plus connue

Premier pas avec Angular

Activité :

- Installer Angular-CLI

Grace à NPM :

```
npm install -g @angular/cli
```

Premier pas avec Angular

Activité :

- Vérifier l'installation d'Angular-Cli

```
$ ng version
```

The Angular logo is displayed in a stylized, monospaced font. It consists of a series of characters and symbols arranged in a grid-like pattern, forming the word 'Angular'.

```
Angular CLI: 16.2.0
```

```
Node: 18.15.0
```

```
Package Manager: npm 9.5.0
```

Premier pas avec Angular

Nous allons générer notre premier projet Angular

```
$ ng new pokemon-app --minimal --style=css
```

Le nom du projet = pokemon-app

le type de fichier de style = css

Premier pas avec Angular

Activité :

Créer notre projet Angular

```
$ ng new pokemon-app --minimal --style=css
```


Premier pas avec Angular

```
$ ng new ng-pokemon-app --minimal --style=css
? Would you like to add Angular routing? Yes
CREATE ng-pokemon-app/angular.json (2989 bytes)
CREATE ng-pokemon-app/package.json (790 bytes)
CREATE ng-pokemon-app/README.md (1066 bytes)
CREATE ng-pokemon-app/tsconfig.json (901 bytes)
CREATE ng-pokemon-app/.gitignore (548 bytes)
CREATE ng-pokemon-app/tsconfig.app.json (263 bytes)
CREATE ng-pokemon-app/.vscode/extensions.json (130 bytes)
CREATE ng-pokemon-app/.vscode/launch.json (297 bytes)
CREATE ng-pokemon-app/.vscode/tasks.json (531 bytes)
CREATE ng-pokemon-app/src/main.ts (214 bytes)
CREATE ng-pokemon-app/src/favicon.ico (948 bytes)
CREATE ng-pokemon-app/src/index.html (298 bytes)
CREATE ng-pokemon-app/src/styles.css (80 bytes)
CREATE ng-pokemon-app/src/app/app-routing.module.ts (245 bytes)
CREATE ng-pokemon-app/src/app/app.module.ts (393 bytes)
CREATE ng-pokemon-app/src/app/app.component.ts (1489 bytes)
CREATE ng-pokemon-app/src/assets/.gitkeep (0 bytes)
✓ Packages installed successfully
```

Premier pas avec Angular

Que fait "Ng New"

- Il crée tout le socle d'Angular
- Il installe toute la dépendance d'Angular (Il effectue des simple NPM INSTALL)
- Il initialise un .git
- Setup des configurations pour des IDE

Présentation du socle

Premier pas avec Angular

Lancé Angular :

```
$ ng server
? Would you like to share pseudonymous usage data about this project
at Google under Google's Privacy Policy at https://policies.google.co
details and how to change this setting, see https://angular.io/analyt
Global setting: enabled
Local setting: disabled
Effective status: disabled
✓ Browser application bundle generation complete.
```

Initial Chunk Files	Names	Raw Size
vendor.js	vendor	2.33 MB
polyfills.js	polyfills	333.19 kB
styles.css, styles.js	styles	230.46 kB

Premier pas avec Angular

Inspection premier composant

Inspection du module racine

Configuration TypeScript

```
8: "strict": true,  
9: "strictPropertyInitialization": false,
```

Composants

Quest-ce qu'un composants ? :

Un composants est une parti de l'écran

Cette portion de l'écran qu'on vas controller,
on appel ça une vue (ligne 5)

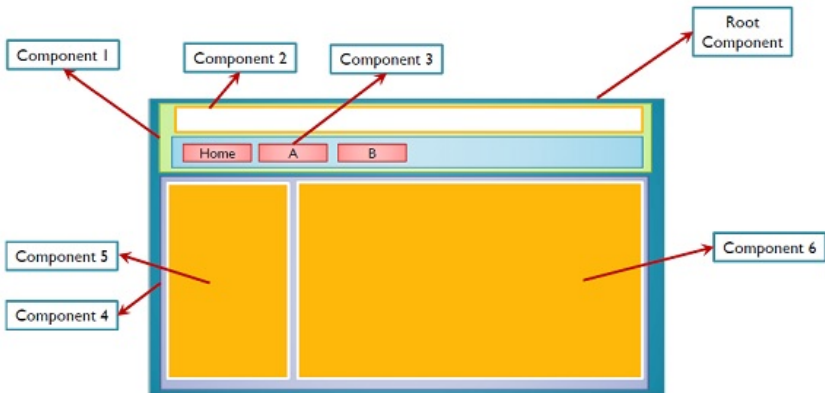
cette vue est deffini dans le template et ça peut-etre beaucoup chose

Donc un composant est une class + une vue

Composants

Qu'est-ce qu'un composant ? :

le template (la vue) c'est le rendu pour les utilisateurs



Composants

Première variable :

```
export class AppComponent {  
  pokemons = ['Moustillons', 'Salameche', 'tortank'];  
}
```

Activité :

Afficher un pokémon dans votre template !

Composants

Affichage de notre première variable :

```
template: `

# Welcome to {{ pokemons[1] }}! </h1>`


```

Activité :

- Créer deux variable de type number
- Mettez deux nombre aléatoire
- Afficher dans le template le resultat de l'addition de ces deux nombre

Information

Pensez a typé vos variable

Composants

Cycle de vie :

ngOnChanges	C'est la méthode appelée en premier lors de la création d'un composant, avant même <i>ngOnInit</i> , et à chaque fois que Angular détecte que les valeurs d'une propriété du composant sont modifiées. La méthode reçoit en paramètre un objet représentant les valeurs actuelles et les valeurs précédentes disponibles pour ce composant.
ngOnInit	Cette méthode est appelée juste après le premier appel à <i>ngOnChanges</i> , et elle initialise le composant après que Angular ait initialisé les propriétés du composant.
ngDoCheck	On peut implémenter cette interface pour étendre le comportement par défaut de la méthode <i>ngOnChanges</i> , afin de pouvoir détecter et agir sur des changements qu'Angular ne peut pas détecter par lui-même.
ngAfterViewInit	Cette méthode est appelée juste après la mise en place de la vue d'un composant (et des vues de ses composants fils s'il en a).
ngOnDestroy	Appelée en dernier, cette méthode est appelée avant qu'Angular ne détruise et ne retire du DOM le composant. Cela peut se produire lorsqu'un utilisateur navigue d'un composant à un autre par exemple. Afin d'éviter les fuites de mémoire, c'est dans cette méthode que nous effectuerons un certain nombre d'opérations afin de laisser l'application « propre » (nous détacherons les gestionnaires d'événements par exemple).

NgOnInit :

On va en chargeant ce composant
vouloir afficher le tableau pokemon dans un console.log

On va devoir importer la fonctionnalité nécessaire

Composants

NgOnInit :

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


Composants

NgOnInit :

```
export class AppComponent implements OnInit {  
  pokemons = ['Moustillons', 'Salameche', 'tortank'];  
  ngOnInit() {  
  }  
}
```

Composants

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

@Component({
  selector: 'app-root',
  template: `<h1> Welcome to {{ pokemonList[1] }}! </h1>`
})
export class AppComponent implements OnInit {
  pokemonList = ['Moustillons', 'Salameche', 'tortank'];

  ngOnInit(){

    console.table(this.pokemonList);

  }
}
```

Activité :

- Créer une alert (Function JavaScript) au démarrage
- Cette alert dira "bonjour" au Utilisateur
- Puis faite un "console.log" d'une nouvelle variable de type String
- La variable dira "Vous n'est pas polie"
- Afficher dans un console.log l'additions de l'activité precedente

Interaction Utilisateur :

On va simplement créer une fonction qui affiche notre pokémon dans la console

Une fois cette fonction créée, on verra dans le template comment activer cette fonction

Composants

Interaction Utilisateur :

```
export class AppComponent implements OnInit {  
  pokemonList = ['Moustillons', 'Salameche', 'tortank'];  
  
  ngOnInit(){  
    console.table(this.pokemonList);  
  }  
  
  selectPokemon(pokemonName: string) {  
    console.log(`vous avez cliqué sur le pokémon ${pokemonName}`)  
  }  
}
```

Composants

Interaction Utilisateur :

Pour tester cette fonction sans le template, on va lancer la fonction au lancement du composant

```
ngOnInit(){  
  console.table(this.pokemonList);  
  this.selectPokemon('Salameche');  
}
```

Activité :

- Créer simplement une fonction "showId"
- qui aura comme argument "id" de type string
- et qui fera simplement un `console.log d'"ID"`

Activité :

- Créer une fonction "multiplication"
- Cette fonction besoin de deux variable number pour foncitionner
- Cette fonction doit multiplier les deux variable et l'afficher dans un console.log
- Créer cette fonction et lancé la fonciton au démarrage du composant en utilisant les deux variable "data1" et "data2"

Gerez de la donnée :

nous allons créer deux fichier

- ./src/app/pokemons.ts
- ./src/app/mock-pokemons-list.ts

Le premier est un model (Class D'Object en php)

Le Deuxième un tableau basé sur le model qui contient de la donnée

Activité :

- Afficher en titre " Liste de pokémon"
- Recupéré dans la pokemonList toute les donnée du mock
- Dans la fonction selectPokemon recupéré tout un model pokémon plutot qu'un String

Laisser la vue au même endroit que la logique
C'est absolument pas **recommander**

Alors dans notre composant on vas changer ça,
en créant un fichier appart

Template

On vas créer un fichier :

- ./src/app/app.component.html

et modifier le composant :

```
@Component({  
  selector: 'app-root',  
  templateUrl: 'app.component.html'  
})
```

Dans app.component.html :

```
<h1>Liste des pokémon</h1>  
<p>Paragraphe</p>  
<p>Paragraphe</p>  
<p>Paragraphe</p>
```

On peut constater que la vue se charge bien par le biais de notre fichier html

Et il y a bien toujours notre logique dans la console JavaScript

Donc maintenant on a un fichier **Logique** (Composant) et un fichier **Graphique** (Template)

c'est ce système qu'on va utiliser tout au long du développement Angular

L'interpolation :

Nous avons actuellement un fichier HTML **static**

Et nous voulons le rendre **dynamique**

On vas utiliser la syntaxe d'interpolation

Template

L'interpolation :

```
<h1>Liste des pokémon</h1>
<p>{{ pokemonList[0] }}</p>
<p>{{ pokemonList[1] }}</p>
<p>{{ pokemonList[2] }}</p>
<p>{{ pokemonList[3] }}</p>
<p>{{ pokemonList[4] }}</p>
<p>{{ pokemonList[5] }}</p>
<p>{{ pokemonList[6] }}</p>
<p>{{ pokemonList[7] }}</p>
<p>{{ pokemonList[8] }}</p>
<p>{{ pokemonList[9] }}</p>
<p>{{ pokemonList[10] }}</p>
<p>{{ pokemonList[11] }}</p>
```


Activité :

- Regler le souci !

Template

L'interpolation :

```
<h1>Liste des pokémon</h1>
<p>{{ pokemonList[0].name }}</p>
<p>{{ pokemonList[1].name }}</p>
<p>{{ pokemonList[2].name }}</p>
<p>{{ pokemonList[3].name }}</p>
<p>{{ pokemonList[4].name }}</p>
<p>{{ pokemonList[5].name }}</p>
<p>{{ pokemonList[6].name }}</p>
<p>{{ pokemonList[7].name }}</p>
<p>{{ pokemonList[8].name }}</p>
<p>{{ pokemonList[9].name }}</p>
<p>{{ pokemonList[10].name }}</p>
<p>{{ pokemonList[11].name }}</p>
```

Activité :

- Créé un tableau nommé "Personne"
 - Dans ce tableau on aura un nom et un prénom et age
 - Puis afficher
- dans le template avec la balise article le Nom, le Prenom et l'Age du tableau

Name : Michel Dupond

Age : 21

Interaction Utilisateur :

Pour l'instant tout ce qu'on a vue c'est de transmettre des donnée de la
Logique à la Vue

mais ce qu'on vas vouloir faire c'est l'inverse

Donc ce qu'on vas vouloir faire :

C'est quand on clic sur un nom de pokemon
excuter la fonction qu'on a créer "selectPokemon"

Interaction Utilisateur :

Donc comment on fait on vas simplement prendre un paragraphe d'un pokémon et on vas lui ajouter ceci

```
<p (click)="selectPokemon(pokemonList[0])">{{ pokemonList[0].name }}</p>
```

Template

Interaction Utilisateur :

On vas vouloir aller plus loin en créant un input qui envoie la donnée a notre fonction "showId"

```
<input
  #input
  (keyup)="showId(input.value)"
  type="number"

/>
```

Tips

En fesant (keyup.enter) la donnée s'envoie uniquement en cliquant sur la touche entré

Activité :

Je veut quand je remplie le formulaire par un id :

- Que la console me sorte le nom du pokémon de cette id
- Et qu'il stock l'object de ce pokémon dans une nouvelle variable "pokemonSelected"

Information

Vous aurez besoin de créer une nouvelle fonction "showPokémon"

NgIf et NgFor :

- NgIf permet d'effectuer une conditions dans le template
- NgFor permet de bouclé l'intergralité d'un tableau (foreach)

NgIf :

On vas afficher un message différent si Salameche est trouver

```
<p *ngIf="pokemonSelected == pokemonList[1]">  
  Vous êtes sur salameche  
</p>
```

NgFor :

On vas vouloir afficher tout les pokémon en quelque ligne









```
<p *ngFor="let pokemon of pokemonList">  
  {{ pokemon.name }}  
</p>
```

Template

Activité :

Je veut cette interface grace au ngFor

Et quand on click sur un pokémon je veut qu'il affiche le nom dans la console

			
Bulbizarre Crée : Sat Aug 26 2023 04:47:42 GMT+0200 (heure d'été d'Europe centrale)	Salamèche Crée : Sat Aug 26 2023 04:47:42 GMT+0200 (heure d'été d'Europe centrale)	Carapuce Crée : Sat Aug 26 2023 04:47:42 GMT+0200 (heure d'été d'Europe centrale)	Aspicot Crée : Sat Aug 26 2023 04:47:42 GMT+0200 (heure d'été d'Europe centrale)
			

Utiliser Bootstrap