

# Module Angular

## Parti 1

Maxime Tournier

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

Angular est un **FrameWork**

## Traduction

Framework = Cadre de travail

En tant que développeur on fait souvent la meme chose

Exemple:

Valider les formulaires — Gerez la navigation — traiter les erreurs

Et pour regler ce problème au lieu d'allez recupere les fonction d'autre projet on à créer les framework

Avantage : Tout le monde travaille sur les même fonction

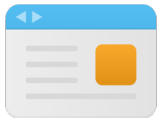
Pour comprendre le fonctionnement d'angular

Il faut comprendre le web aujourd'hui

Site Web / Application Web

Il s'agit bien de deux chose différente

Avant ça comment ça marche :



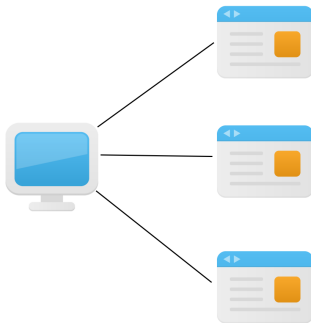
- Fichier client (CSS, HTML, JS)



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

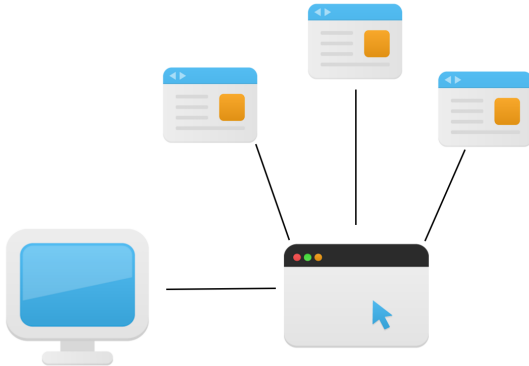
# Présentation

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 deleguer le travaille a JavaScript qui vas désactivé ou activé une parti du code html/css

## Information

Cette méthode s'appel SPA (Single Page Application)



AngularJS n'est pas égal a Angular

AngularJS à été créer 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 codé une multitude de petit composant qui formeront une application

Un composant est une parti du site qui fonctionne de manière autonome sur une applicaiton

Angular est il difficile à apprendre ? :

Angular à une réputation de framework difficile

Car Angular est basé sur  
TypeScript un langage basé sur Javascript qui permet de Typé vos variable

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 developpement
2. Genere un socle angular (Grâce à Angular-CLI)
3. Nous allons jouer avec le composant racine =)

## Information

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

# Démarrage Angular

Ce qu'on a besoin :

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

D'un Editeur de code TypeScript

Un outil qu'on appel Angular CLI (Qui vas nous permet de gérez notre projet Angular)

NodeJS et NPM sont utiliser pour la majorité des projet JavaScript moderne

NodeJS permet d'excute du code JS coté serveur

NPM

permet de gere les dépenche de paquet de javascript (angular est un paquet)

NodeJS integre par default 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 vas avoir besoin d'un IDE

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

- Visual Studio Code : Recommander par Microsoft (Gratuit)
- WebStorm : Beaucoup plus puissant et recommandé 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 leur site internet

Pour votre terminal :

Souvent il peut avoir des problème 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 developé 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 :

Activité :

# Premier pas avec Angular

Angular-CLI :

Cette application vas installer tout ce qu'on a besoin  
en une seul ligne de commande

et avec cette même outils on vas pouvoir piloté certaine fonctionnalité  
d'angular directement en ligne de command

## Information

Ce n'est pas la seul méthode pour démarré 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 :

- Verifier l'installation de Angular-CLI

```
$ ng version
```

```

  _/   \_  _/   \_  _/   \_  _/   \_  _/   \_  _/   \_  _/   \_  _/   \_
 /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \ /  _  \
/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \
/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \/_  _  \
      |  _/

```

```
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 toutes les dépendances 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

# Premier pas avec Angular

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

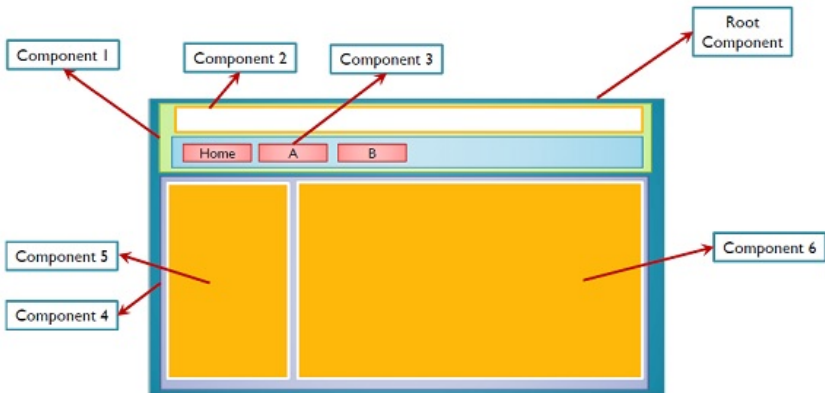
Donc un composant est une class + une vue



# 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 !

Affichage de notre première variable :

```
template: `

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


```

Activité :

Activité :

# Composants

## Cycle de vie :

<b>ngOnChanges</b>	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.
<b>ngOnInit</b>	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.
<b>ngDoCheck</b>	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.
<b>ngAfterViewInit</b>	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).
<b>ngOnDestroy</b>	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 importé la fonctionnalité nessesaire



# 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 :

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 :



Activité :

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

Activité :

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>
```