

Angular

EN VOL 2021

Tifenn Guillas

François Agneray

Tifenn Guillas - François Agneray

Ingénieurs en développement et déploiement
d'application

Laboratoire d'Astrophysique de Marseille - LAM

tifenn.guillas@lam.fr - françois.agneray@lam.fr

5 ans d'expérience sur Angular



Plan

- Angular, qu'est ce que c'est ?
- Quelques concepts
- Angular comment ça marche ?
- Principaux packages
- Testing
- Retour d'expérience
- Et maintenant la pratique



Angular, quésaco ?

Angular, c'est quoi ?

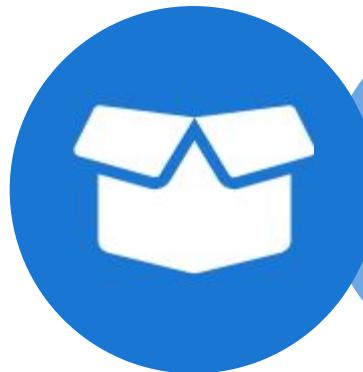
- Framework JavaScript
- Single Page Application
- Orientée composants
- Microsoft TypeScript
- RxJS
- Crée par Google (Licence MIT)
- Première version en septembre 2016
- **Attention** : AngularJS != Angular



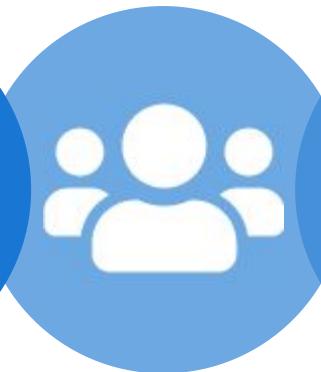
In Angular we trust:



Angular en chiffres



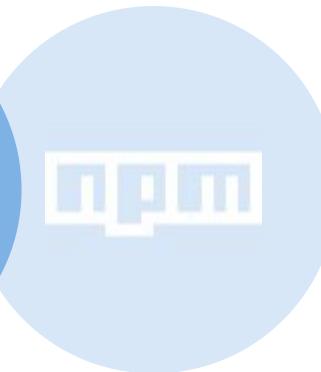
28
packages



1300+
contributeurs



1,7+
projets



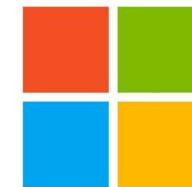
10 000+
addons

Dernière version : 13.0.0 (04 Novembre 2021)

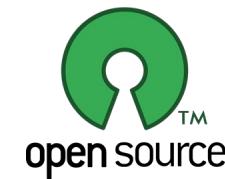
Quelques concepts

TypeScript

- Créé en 2012
- Développé par Microsoft
- Open source
- Typage fort
- Langage orientée objet
- TS = JS + Types + Classes + Interfaces + Imports...
- Nécessite une compilation (**tsc**)
- Extension de fichier : **.ts**



Microsoft



TypeScript - Typage statique

```
// Création d'une variable contenant une valeur booléenne.  
let myBool: boolean = false;  
  
// Création d'une variable contenant une chaîne de caractère.  
let myString: string = "Hello World!";  
  
// Création d'une variable contenant un nombre.  
let myNumber: number = 1;  
  
// Création d'une fonction renvoyant une chaîne de caractère.  
function myFunction(): string {  
    return "Say cheese =D";  
}
```

TypeScript - Type générique

```
function myFunction<T>(parameter: T) {  
    // Contenu de la fonction.  
}  
  
class MyClass<T> {  
    myVar: T;  
    // Contenu de la classe.  
}  
  
// Création d'une instance de la classe "MyClass" en définissant un type.  
let myInstance = new MyClass<string>();  
myInstance.myVar = "Hello World!";
```

TypeScript - Interfaces

```
interface MyInterface {  
    // Création d'une signature de variable.  
    myVar: string;  
    // Création d'une signature de méthode.  
    myMethod(parameter: string): void;  
}  
  
class MyClass implements MyInterface {  
    myVar: string;  
    myMethod(parameter: string): void {  
        // Contenu de la méthode.  
    }  
}  
  
// Précision du type de la variable en utilisant l'interface.  
let instance: MyInterface = new MyClass();
```

TypeScript - Classes

```
class MyClass {  
    private _firstname;  
    private _lastname;  
  
    public constructor(firstname: string, lastname: string) {  
        this._firstname = firstname;  
        this._lastname = lastname;  
    }  
  
    public sayHi(): string {  
        return "Hi " + this._firstname + " " + this._lastname;  
    }  
}  
  
// Création d'une instance de "MyClass"  
let myInstance: MyClass = new MyClass("Jean-Claude", "Van Damme");  
myInstance.sayHi();
```

TypeScript - Héritage

```
// La classe hérite de "MyClass".  
  
class MyNewClass extends MyClass {  
    public constructor(firstname: string, lastname: string) {  
        // Accède au constructeur de "MyClass".  
        super(firstname, lastname);  
    }  
}  
  
// Création d'une instance de "MyNewClass" et  
// appel de la méthode: "sayHi" de la classe parente : "MyClass".  
  
let myInstance: MyNewClass = new MyNewClass("Jean", "Dupond");  
myInstance.sayHi();
```

TypeScript - Compilation

- Nécessite une compilation pour fonctionner
- TypeScript fournit un compilateur (tsc)
- Processus de compilation intégré à Angular (JS, HTML, CSS)



RxJS

- Programmation réactive
- Utilise des streams (flux de données)
- Remplace les promesses dans Angular
- Utilise des *Observables* :
 - Gestion des évènements asynchrones
 - Valeurs multiples
 - Similaire publish/subscribe pattern
- Permet la gestion des événements de la page
 - Clic sur la page
 - Chargement de la page
 - ...
- Opérateurs (`.map()`, `.reduce()`, `.forEach()` ...)
- Gestion des réponses serveur dans Angular



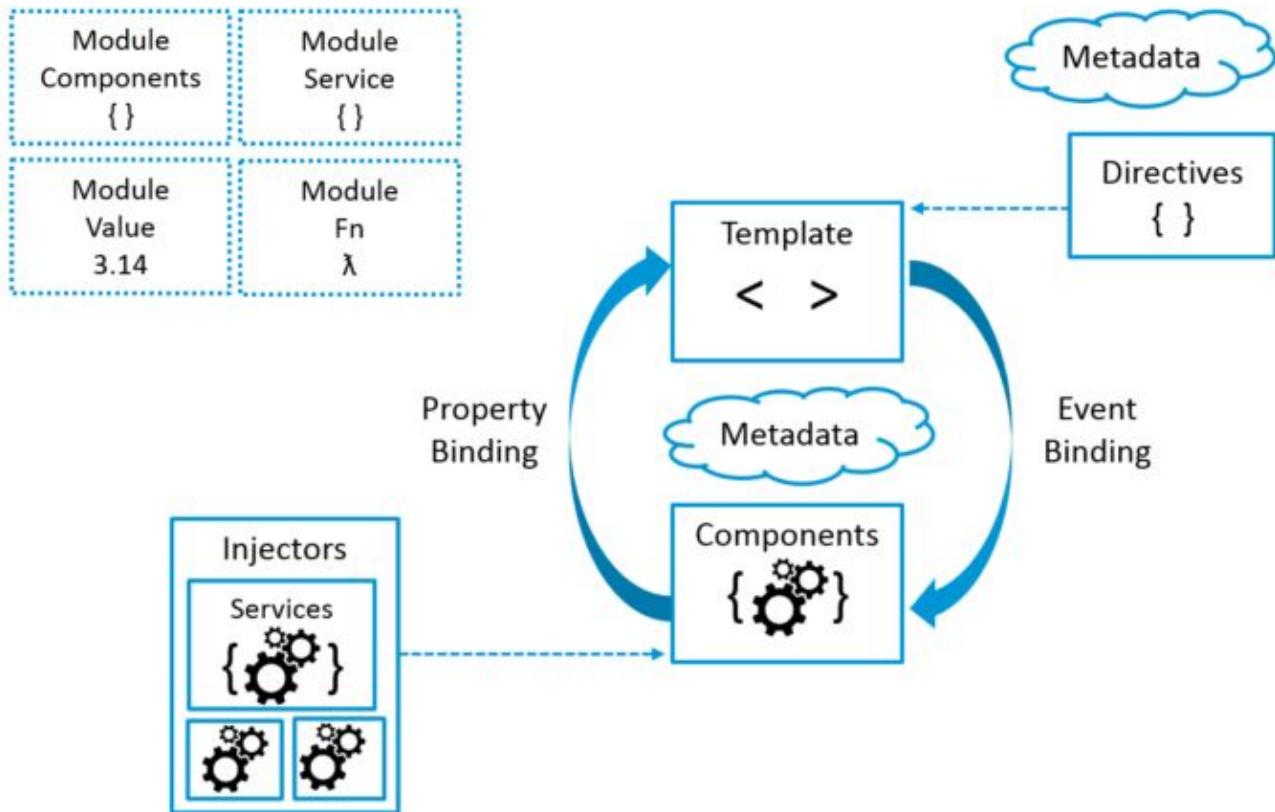
RxJS - Exemple Angular HttpClient

- HttpClient fournit des méthodes pour interroger un serveur web
- Appels HTTP asynchrones
- Les réponses sont sous la forme d'Observables RxJS (stream)
- Possibilité de modifier et filtrer les données à la volée (pipe operator)
- `.subscribe` lance une fonction de callback sur la réponse

```
ngOnInit() {  
    this.http.get<Article[]>("/server/articles") .pipe(  
        filter(article => article.price > 10)  
    ) .subscribe(data => {  
        this.articles = data;  
    }) ;  
}
```

Angular comment ça marche ?

Architecture

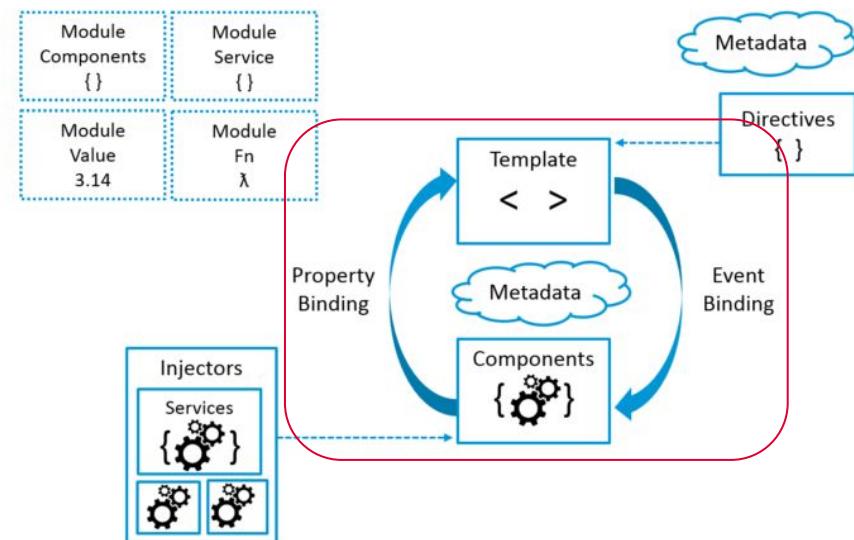


Component

- Contrôleur : Classe TypeScript
- Vue : HTML, CSS
- Data binding
- Event binding

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  recipe: string = 'raclette';
  // logic here
}
```

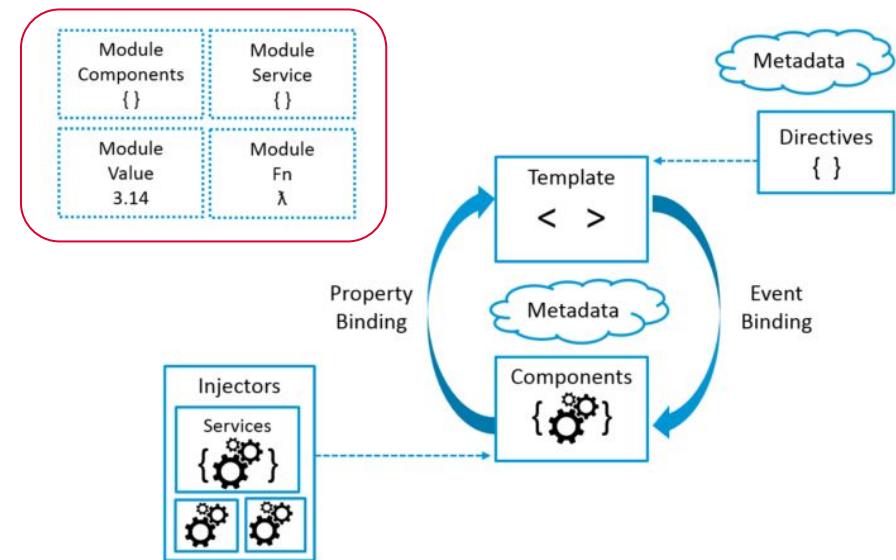
```
<h1>{{ recipe }}</h1>
<input [(ngModel)]="recipe">
```



Module

- Applications modulaires
- Root module : *AppModule*
- NPM package = module

```
@NgModule({  
    declarations: [AppComponent],  
    imports: [BrowserModule],  
    exports: [],  
    providers: [],  
    bootstrap: [AppComponent]  
})  
export class AppModule { }
```

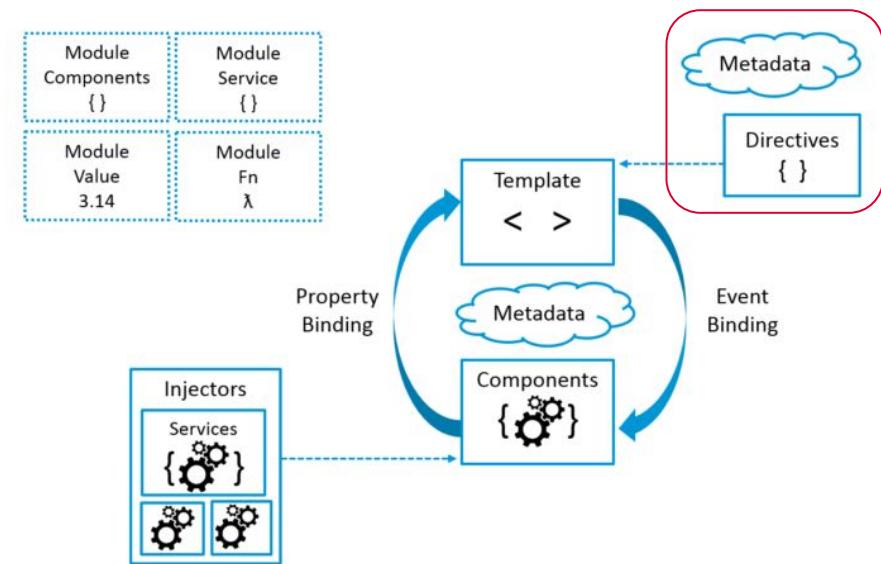


Directive

- Transformation du DOM
- `@Directive`
- Deux types :
 - Directive structurelle (`*ngIf`, `*ngFor`, `*ngSwitch`)
 - Directive d'attribut (`ngClass`, `ngStyle`)

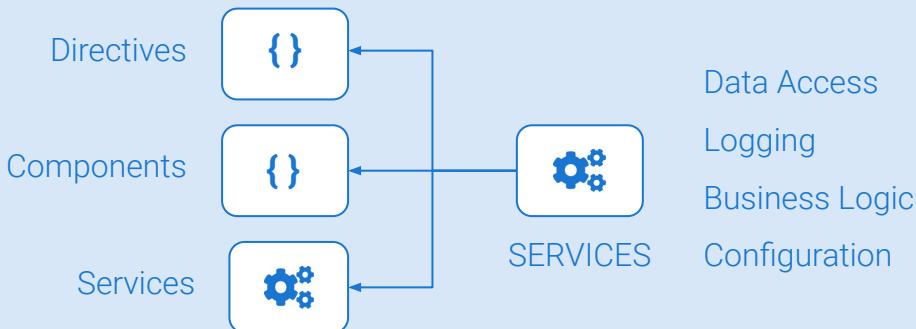
```
<div *ngIf="recipe">{{ recipe }}</div>
```

```
<div [ngClass]="{{ 'text-success' : true }}>
  Recipe saved!
</div>
```

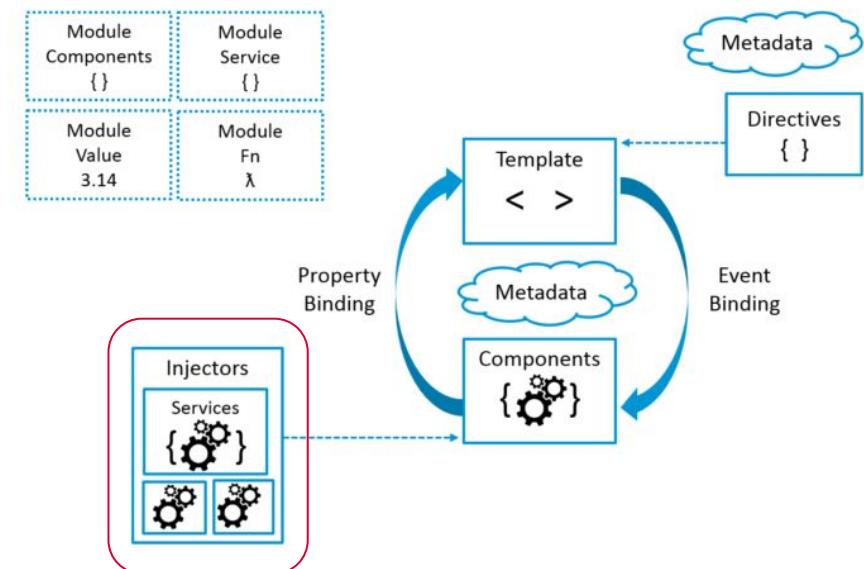


Service

- Service = dépendance
- Injection de dépendances



```
@Injectable()
export class FetchDataService { }
```



Les principaux packages

Ready to rocks!

Inclu :

- Core & Common
- CLI
- Router
- HttpClient
- Forms
- Karma
- Animations



Optionnel :

- I18n
- Material
- Universal



Extension navigateur :

- Augury



Plugin IDE :

- Angular Language Service



Angular CLI



```
alphonse:~$ ng new my-revolutionary-app
```

```
alphonse:~$ ng add bootstrap
```

```
alphonse:~$ ng generate module
```

```
alphonse:~$ ng generate service
```

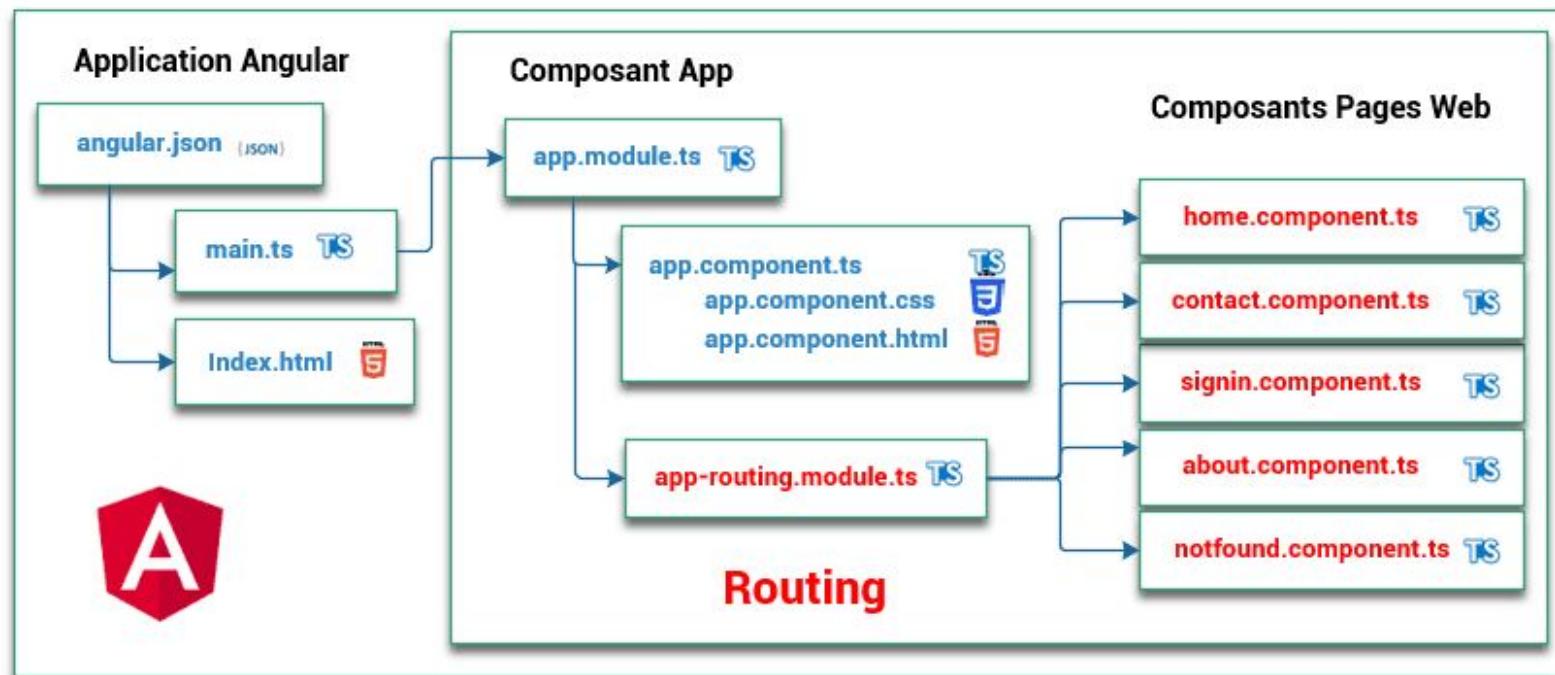
```
alphonse:~$ ng generate component
```

```
alphonse:~$ ng serve
```

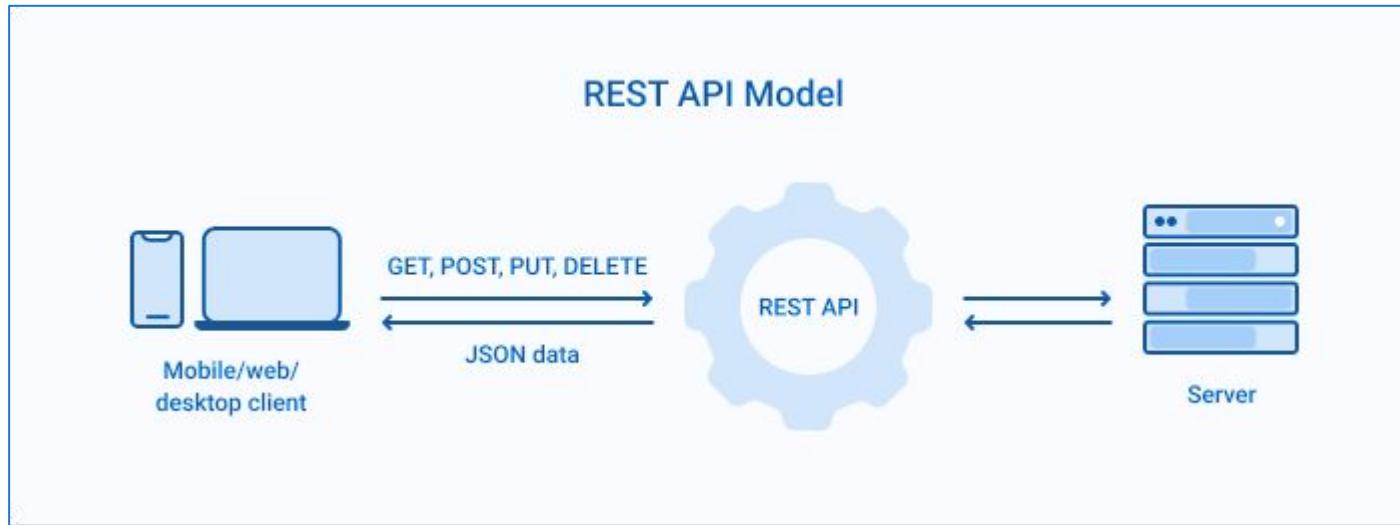
```
alphonse:~$ ng test
```

```
alphonse:~$ ng build
```

Router



HttpClient



Forms



Reactive

```
// BoringFormComponent.ts
@Component({
  selector: 'app-boring-form',
  templateUrl: `boring-form.component.html`
})
export class BoringFormComponent {
  nameControl = new FormControl('');
}
```

```
<!-- boring-form.component.html -->
Name: <input [formControl]="nameControl">
```

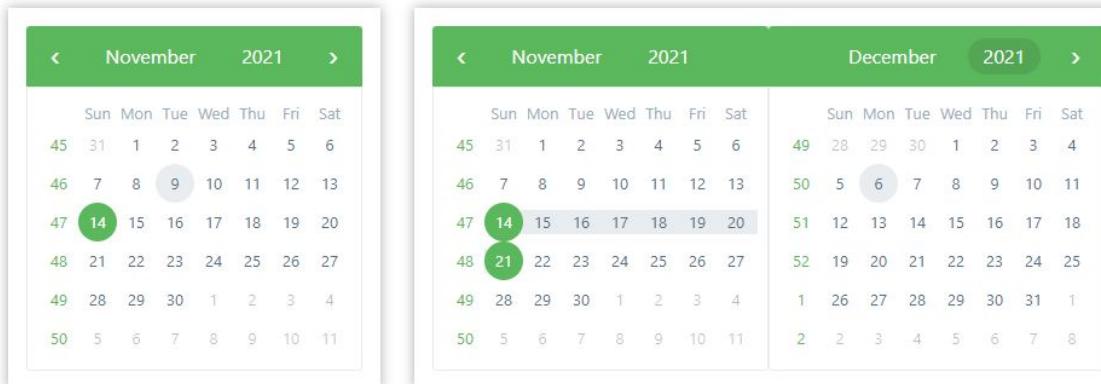
Template-driven

```
// BoringFormComponent.ts
@Component({
  selector: 'app-boring-form',
  templateUrl: `boring-form.component.html`
})
export class BoringFormComponent {
  name: string = '';
}
```

```
<!-- boring-form.component.html -->
Name: <input [(ngModel)]="name">
```

ngx-bootstrap

- Composants Bootstrap 3, 4 ou 5 pour Angular
- Simple d'utilisation
- Documentation et communauté
- Composants de base disponibles (accordéon, carrousel, collapse, datepicker...)



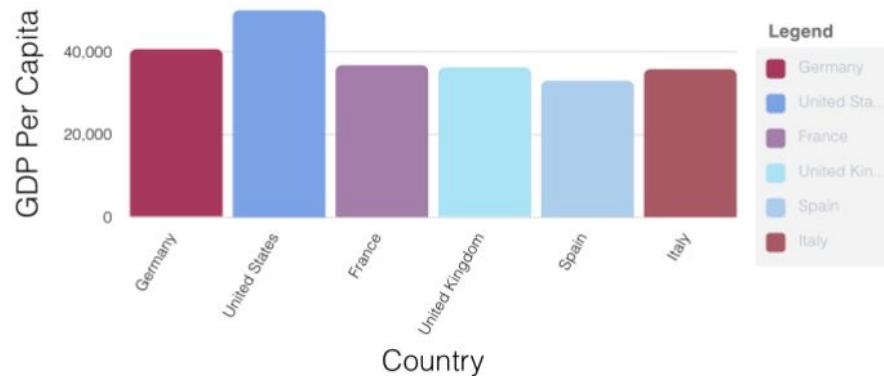
ngx-toastr

- Permet l'affichage de notifications



ngx-charts

- Permet de créer des graphiques (Horizontal & Vertical Bar, Line, Area, Pie...)



Testing

"Tu peux pas test !"

- Tests unitaire
 - Karma + Jasmine
 - Jest
 - Extension de fichier : **.spec.ts**
- Tests fonctionnels
 - Cypress
 - Nightwatch.js
 - WebdriverIO



Exemple d'un test

```
// app.components.spec.ts
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  let app: AppComponent;

  beforeEach(() => {
    // Configuration de l'environnement d'exécution des tests
    TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
    }).compileComponents();
    const fixture = TestBed.createComponent(AppComponent);
    app = fixture.componentInstance;
  });

  it('should create the app', () => {
    expect(app).toBeTruthy();
  });
});
```

Retour d'expérience

Pros - Cons

- + Expérience utilisateur
- + Réactivité
- + Framework complet
- + Communauté
- + Développement rapide (!!)
- + Fait par Google
- Temps d'apprentissage
- Mise à jour régulière
- SEO
- Clivage parmi les devs

Sources

<https://angular.io/>

<https://github.com/angular/angular>

<https://www.npmjs.com/package/@angular/core>

<https://www.typescriptlang.org>

<https://angular.io/cli>

<https://www.udemy.com/course/the-complete-guide-to-angular-2>

<https://www.edureka.co/blog/angular-tutorial>

<https://gitlab.lam.fr/anis>

Et maintenant la pratique

Le TP

Disponible sur :

- La machine virtuelle
- <https://cesam.lam.fr/envol-2021-tp-angular/>

Deux parties :

- Apprentissage du Framework avec des exemples simples
- Un exemple plus complexe utilisant une API de <https://geo.api.gouv.fr>

