

ANGULAR

Objectifs

- ☑ Créer des composants angular
- ☑ Manipuler un template avec le databinding, les directives et les pipes
- ☑ Faire communiquer les composants avec les services et les Observables
- ☑ Créer des Single Page Applications avec le routing
- ☑ Communiquer avec les utilisateurs avec les formulaires
- ☑ Faire communiquer une application avec un backend

Historique des frameworks

Aux débuts du développement web, seul le HTML permet de créer des sites.

- En 1990, les sites ressemblent à des documents texte, car il n'existe pas vraiment d'autres possibilités.
- En 1998, le CSS arrive pour aider à créer des mises en page plus esthétiques.
- En 2000, le JavaScript commence son règne sur le développement web, permettant des interactions entre l'utilisateur et la page.
- En 2005, le système AJAX (Asynchronous Javascript And XML) permet des interactions entre l'utilisateur et des backend HTTP : il est enfin possible d'échanger des informations et de générer du contenu à partir de ces interactions.

Historique des frameworks

- En 2010, la première version d'AngularJS est lancée.

Elle permet de créer plus facilement des Single Page Applications (SPA), des applications web qui imitent les applications natives : pas de rafraîchissement du navigateur, temps de chargement réduits, une UI beaucoup moins “internet” etc.

Cette version permet déjà de faire énormément de choses, mais souffre d'une syntaxe plutôt complexe ainsi que des limitations du JavaScript. Voilà pourquoi Google choisit de complètement réécrire le framework pour sa version 2.

Aujourd'hui, nous en sommes à Angular6.x (maintenant appelé simplement “Angular”) ; la version 3 ayant été sautée pour des raisons sémantiques tout simplement.

Pourquoi Angular ?



Vue.js



marko



ember



Sencha



BACKBONE.JS



React



aurelia

Pourquoi Angular ?

Il y a plusieurs frameworks JavaScript très populaires aujourd'hui : Angular, React, Ember, Vue... les autres frameworks marchent très bien, ont beaucoup de succès et sont utilisés sur des sites extrêmement bien fréquentés, React et Vue notamment.

Angular présente également un niveau de difficulté légèrement supérieur, car on utilise le TypeScript plutôt que JavaScript pur ou le mélange JS / HTML de React.

Pourquoi Angular ?

Quels sont donc les avantages d'Angular ?

- ☑ Angular est géré par Google — il y a donc peu de chances qu'il disparaisse, et l'équipe de développement du framework est excellente.
- ☑ Le TypeScript — ce langage permet un développement beaucoup plus stable, rapide et facile.
- ☑ Le framework Ionic — le framework permettant le développement d'applications mobiles multi-plateformes à partir d'une seule base de code — utilise Angular.

Les autres frameworks ont leurs avantages également, mais Angular est un choix très pertinent pour le développement frontend.

Qu'est-ce que le TypeScript ?

Pour faire bref, le TypeScript est un sur-ensemble (un “superset”) de JavaScript qui est justement transcompilé (transcompilation : "traduction" d'un langage de programmation vers un autre - différent de la compilation, qui transforme généralement le code vers un format exécutable) en JavaScript pour être compréhensible par les navigateurs.

Il ajoute des fonctionnalités extrêmement utiles, comme, entre autres :

- ☑ Le typage strict, qui permet de s'assurer qu'une variable ou une valeur passée vers ou retournée par une fonction soit du type prévu ;
- ☑ Les fonctions dites lambda ou arrow, permettant un code plus lisible et donc plus simple à maintenir ;
- ☑ Les classes et interfaces, permettant de coder de manière beaucoup plus modulaire et robuste.

Environnement de développement

Qu'est-ce que le CLI ?

Le CLI, ou “Command Line Interface” (un outil permettant d'exécuter des commandes depuis la console), d'Angular est l'outil qui vous permet d'exécuter des scripts pour la création, la structuration et la production d'une application Angular.

Pour accéder à la ligne de commande :

- ☑ Sous Windows : utilisez l'invite de commande (Windows+R => CMD). Sinon, il existe des utilitaires natifs comme PowerShell ou des tiers comme PowerCmd ou ConEmu par exemple. Toutes les commandes citées dans le cours devraient fonctionner si vous êtes authentifié comme administrateur sur votre machine.
- ☑ Sous MAC : il suffit de lancer Terminal depuis Spotlight ou le Launchpad.
- ☑ Sur Linux : lancez le terminal de votre choix.

Environnement de développement

Pour l'installation suivez les étapes suivantes :

Etape 1 : Installation Node.js

Téléchargez et installez la dernière version LTS de Node.js ici

<https://nodejs.org/en/download/>

Etape 2 : Installation npm

NPM est un package manager qui permet l'installation d'énormément d'outils et de libraries dont vous aurez besoin pour tout type de développement. Pour l'installer, ouvrez une ligne de commande et tapez la commande suivante :

```
npm install -g npm@latest
```

Etape 3 : Installation CLI Angular

Vous allez maintenant installer le CLI d'Angular de manière globale sur votre machine avec la commande suivante (avec sudo si besoin) :

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


CREATION D'UN PROJET

☑ Pour créer un projet, il utilise Angular CLI via la commande « ng » :

- `ng new FirstAngularApp`
- `cd FirstAngularApp`
- `ng serve —open`

Si tout se passe bien, vous verrez les informations du serveur qui se lance à l'adresse *localhost:4200* et votre navigateur préféré se lancera automatiquement avec le message "Welcome to app!!" et le logo Angular.

Félicitations, votre environnement de développement est prêt !

Welcome to app!



Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)



OUTILS

☑️ Installez VS Code

☑️ Nous allons Google Chrome

☑️ Installez bootstrap : *npm install bootstrap@3.3.7 --save*

Cette commande téléchargera Bootstrap et l'intégrera dans le `package.json` du projet. Il vous reste une dernière étape pour l'intégrer à votre projet. Ouvrez le fichier `.angular-cli.json` du dossier source de votre projet. Dans "apps", modifiez l'array `styles` comme suit puis lancez votre projet (*ng serve --open*):

```
"styles": [  
  "../node_modules/bootstrap/dist/css/bootstrap.css",  
  "styles.scss"  
]
```

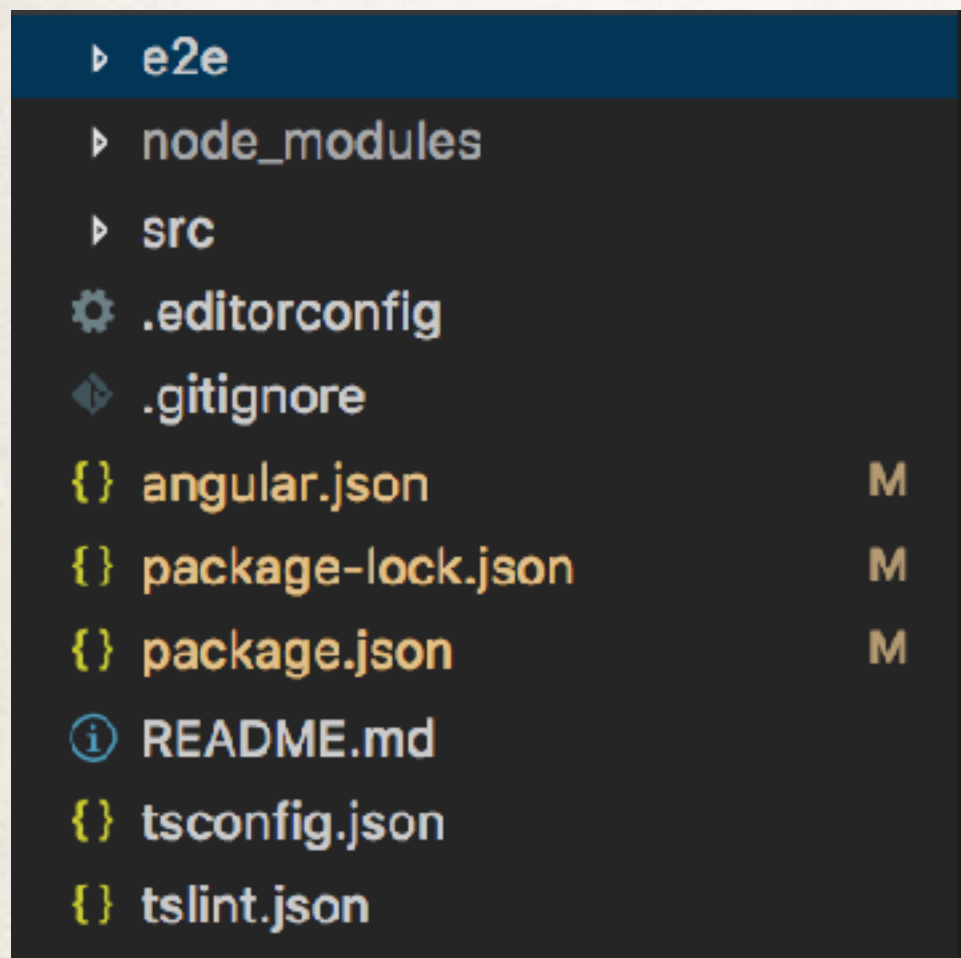

Structuration du projet

Les components sont les éléments de base d'une application Angular : une application est une arborescence de plusieurs components.



Nous avons un component global « AppComponent » tous les autres composants lui sont emboîtés

Structuration du projet



☑ e2e est créé pour les tests unitaires

☑ src - Le principal répertoire de notre projet

☑ node_modules - tous les modules installés sur notre projet

☑ angular.json - le fichier de config d'angular

☑ package.json - Liste toutes les dépendances du projet. Tous les modules qui ont été installé.

☑ tsconfig.json - Définit la configuration du transpileur

☑ tslint.json - Définit des règles de formatage du code

Structuration du projet

Etudions à présent le répertoire src:

- ☑ - main.ts

- ☑ - index.html

- ☑ - Répertoire « app »

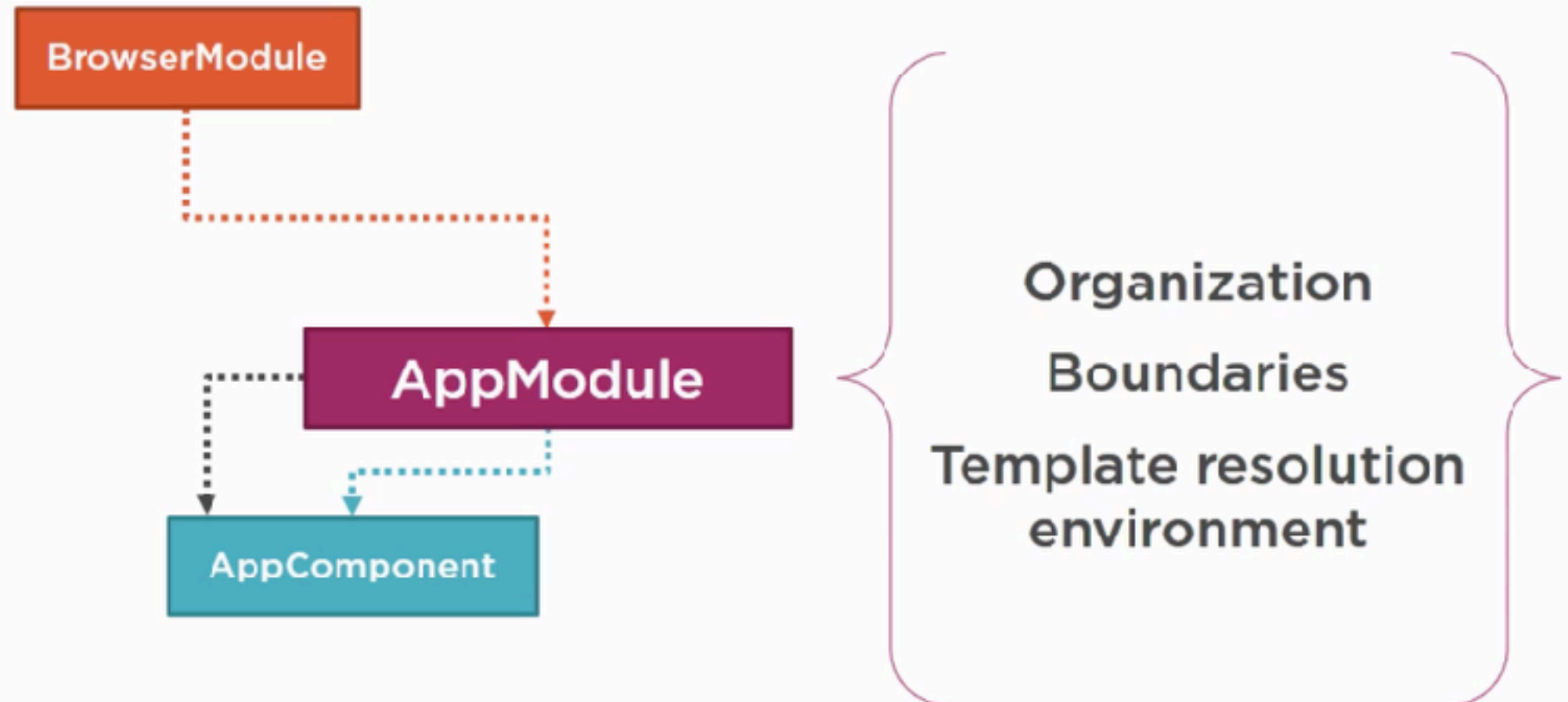
 - ◆ app.module.ts

 - ◆ app.component.ts

 - ◆ app.component.scss

 - ◆ app.component.html

Structuration du projet



- Imports
- Exports
- Declarations
- Providers
- Bootstrap

Création d'un composant

Nous allons créer un nouveau composant à l'aide du CLI d'Angular. Depuis le dossier principal de votre projet, tapez la commande suivante :

==> `ng generate component mon-premier`

```
MacBook-Pro-de-Godwin-2:angularproject godwinavodagbe$ ng generate component firstcompo
CREATE src/app/firstcompo/firstcompo.component.scss (0 bytes)
CREATE src/app/firstcompo/firstcompo.component.html (29 bytes)
CREATE src/app/firstcompo/firstcompo.component.ts (286 bytes)
UPDATE src/app/app.module.ts (412 bytes)
```


Création d'un composant

Le fichier « *app.module.ts* » a également été mis à jour.

Ajoutons à présent ce nouveau composant dans notre page « *app.component.html* »

```
<div style="text-align:center">  
  <h1>  
    Welcome to {{ title }}!  
  </h1>  
  <img width="300" alt="Angular Logo" src="">  
</div>  
<app-firstcompo></app-firstcompo>
```

Databinding

Le databinding, c'est la communication entre le code TypeScript et le template HTML qui est montré à l'utilisateur. Cette communication est divisée en deux directions :

- ☑ Les informations venant de votre code qui doivent être affichées dans le navigateur, comme par exemple des informations que votre code a calculé ou récupéré sur un serveur. Les deux principales méthodes pour cela sont le "string interpolation" et le "property binding" ;
- ☑ Les informations venant du template qui doivent être gérées par le code : l'utilisateur a rempli un formulaire ou cliqué sur un bouton, et il faut réagir et gérer ces événements. On parlera de "event binding" pour cela.

Databinding

Il existe également des situations comme les formulaires, par exemple, où l'on voudra une communication à double sens : on parle donc de "two-way binding ».



Databinding

1- String interpolation

L'interpolation est la manière la plus basique d'émettre des données issues de votre code TypeScript.

DEMO

☑ Créons à présent un nouveau component `AppareilComponent` avec la commande suivante :

==> *ng generate component appareil*

☑ Modifions à présent le fichier « *appareil.component.html* »

```
<li class="list-group-item">
  <h4>Appareil : {{ appareilName }}</h4>
</li>
```

Databinding

☑ Modifions à présent le fichier « *app.component.html* »

```
<div class="container">
  <div class="row">
    <div class="col-xs-12">
      <h2>Mes appareils</h2>
      <ul class="list-group">
        <app-appareil></app-appareil>
        <app-appareil></app-appareil>
        <app-appareil></app-appareil>
      </ul>
    </div>
  </div>
</div>
```

☑ Modifions à présent le fichier « *app.component.ts* »

```
export class AppareilComponent implements OnInit {
  appareilName: string = 'Machine à laver';
```