**AngularJS**

-> Créer des applications web dynamiques et immersives.

-> Créer facilement des Single Page Application

**Single Page Application** = Application Web monopage, application web accessible via une unique page. Cela évite le chargement d’une nouvelle page.

**Avantages**

* géré par Google
* TypeScript : ce langage permet un développement beaucoup plus stable, rapide et facile.
* Le framework Ionic, permettant de développer des applications mobiles à partir d’une seule base de code, utilise AngularJS

**TypeScript**

Superset de JavaScript est transcompilé en JavaScript

Fonctionnalités :

* typage strict ⇒ s’assurer qu’une variable soit du type prévu
* Fonctions lambda ou arrow ⇒ permet un code plus lisible
* Classes et Interfaces ⇒ coder de manière beaucoup plus modulaire et robuste

**Initialisation d’un projet**

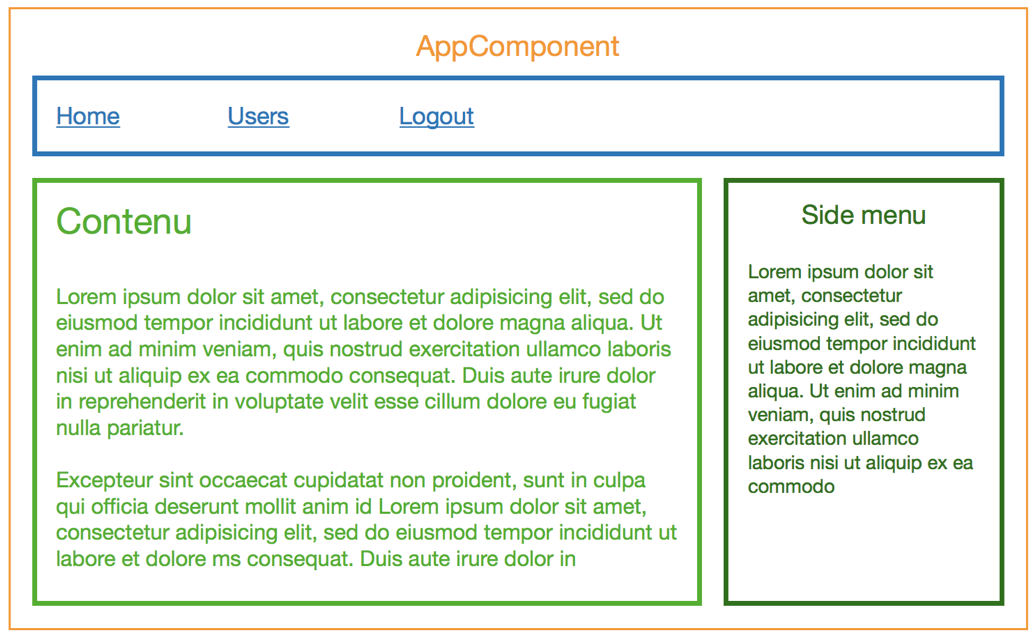
ng new mon-projet-angular --style=scss —skip-tests=true

**Installation de bootstrap**

npm install bootstrap@3.3.7 —save

Ouvrir le fichier angular.json et le modifier tel que :



Un composant pour chaque partie (menu, contenu, side menu).

**Arborescence**

Src/app : contient module principale de l’app + 3 fichiers du compo principal

Présence d’un décorateur @Component()

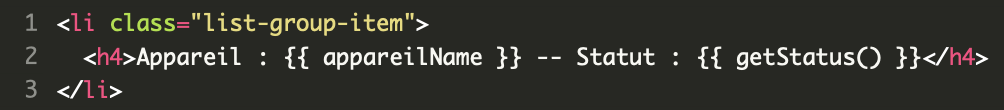
* selector = nom de la balise HTML pour afficher ce composant
* templateUrl = le chemin vers l’HTML à injecter
* styleUrls : chemin vers la ou les feuilles de style de ce component

**Générer un nouveau composant**

ng generate component mon-premier

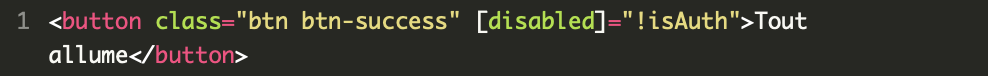
**String interpolation**

Fait d’appeler une variable, le résultat d’une fonction entre deux accolades :



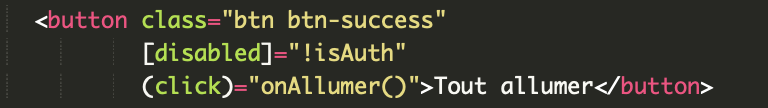
**Property Binding**

Fait de pouvoir changer un attribut d’un DOM à l’aide d’une variable TypeScript.

Symbolisé par des crochets [ ]

**Event Binding**

Fait de réagir aux événements venant du template HTML.

Symbolisé par des parenthèses ( )

**Two-way binding : Liaison à double sens**

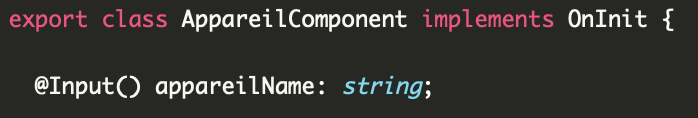
Utilise la liaison par propriété (Property Binding) et la liaison par événement (Event Binding) en même temps.

*Import de FormsModule dans AppModule*

Symbolisé par le mélange crochets - parenthèses : [( )]

On **lie** ici la variable **appareilName** en utilisant la directive **ngModel.**

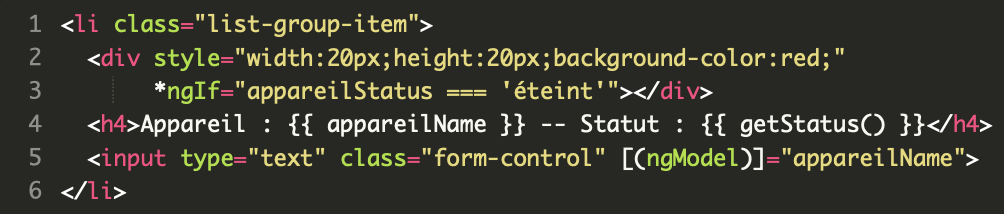
Il faut ensuite appliquer un décorateur à l’attribut appareilName :



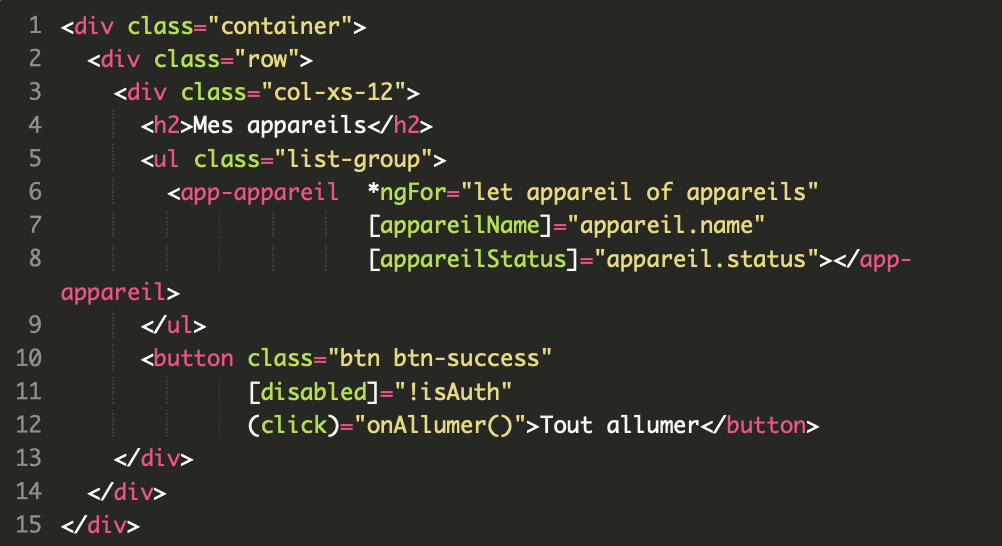
**Les directives**

*Les directives structurelles*

Modifient la structure du document.

ngIf

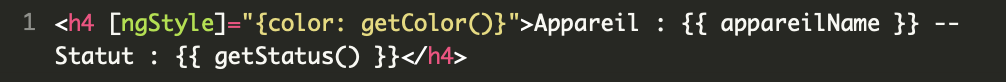
Ici, on affiche un carré rouge si l’appareil est éteint.

ngFor

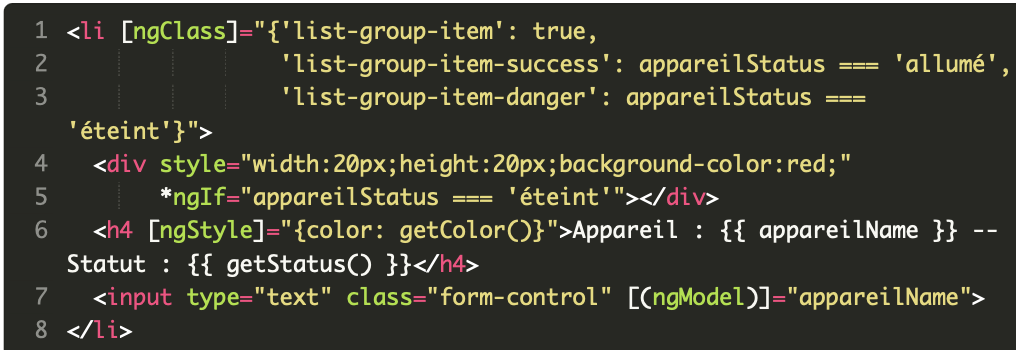
On utilise la boucle for pour créer les app-appareil.

*Les directives par attribut.*

**ngStyle**

**

**ngClass**

****

**Ng-app** : spécifie l’application angular (sur body)

**ng-model** : nom d’une variable

Exemple dans input on spécifie ng-model=‘task’

Et on l’écho comme :

{{task}}

**Ng-repeat** : boucle for

ng-repeat=‘user in users’

**Les filtres**

* modifier les variables
* Filtrer
* Organiser

Pour utiliser un filtrer on utilise un pipe puis le filtre

Exemple : {{user|uppercase}}

**Filter** : prend en paramètre un String, Objet ou fonction

Exemple : {{users | filter:’m’}}

Renverra que les éléments de la liste qui contiennent un m

**orderBy** : +username ou -username

Trie dans l’ordre alphabétique ou - inverse

**Récupérer les composants enfant :**

**@ViewChildren(AppareilComponent) appareils: QueryList<AppareilComponent>;**

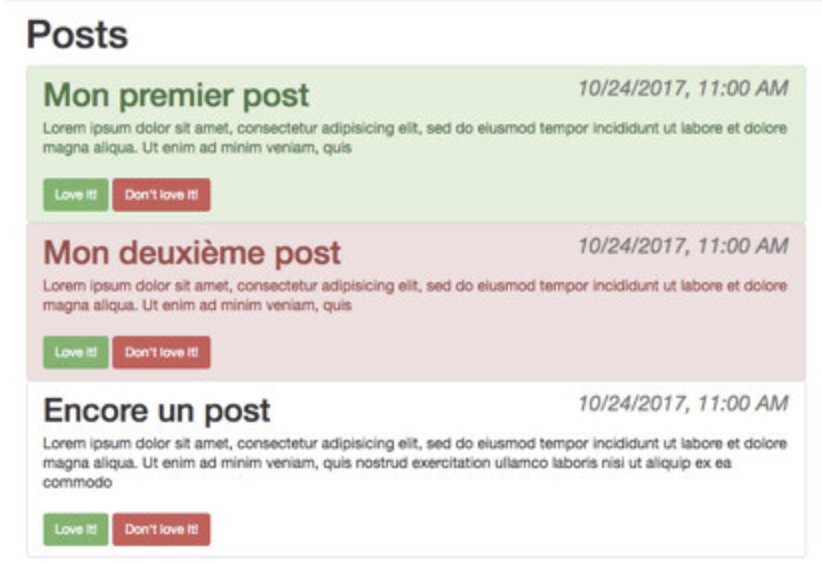
**Les pipes**

-> prennent les données en entrées, les transforment et les affichent.

| date permet de filtrer la donnée au format date

Async permet de spécifier que la donnée vient d’une prémisse et ainsi la filtrer.

**TP Synthèse**

L’objectif est de créer une application telle que :

**Structure d’une classe**

**Les services**

-> centraliser des parties du code et des données qui sont utilisées par plusieurs parties de l’application.

*Injection et instances*

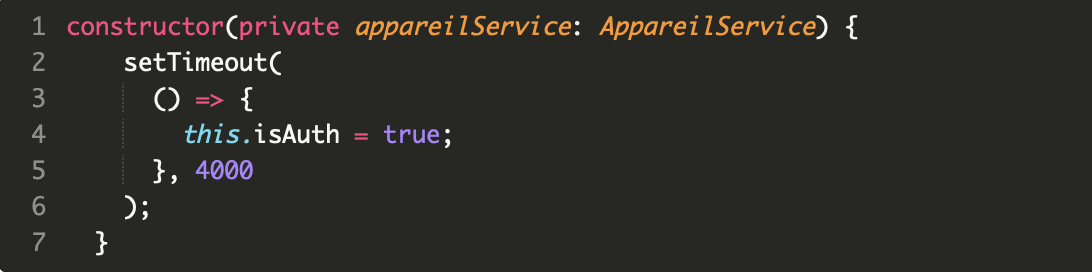
3 niveaux :

* **dans AppModule** : la même instance du service sera utilisée par tous les components de l’application **et par les autres services.**
* **D****ans AppComponent** : tous les composants auront accès à l’instance **mais pas les autres services**
* **Dans un autre Component** : accessibilité par le composant lui même et ses enfants.

**Cas dans AppModule**

-> créer un fichier appareil.service.ts dans un dossier créé dans le dossier app

L’ajouter à l’array providers dans « app.module.ts ».

Pour intégrer ce nouveau service dans l’application, dans AppComponent :

**Utilisation du service**