Notes sur le module : *Angular.*

Armel Pitelet

20 janvier 2022

Table des matières

1 Bases

defiAngularframework (cadre de travail) permettant de faire des applications bureau mais surtout web. Historiquement c'était Angularjs et maintenant Angular 13. Maintenue par google, il est réttocompatible et sécurisé. C'est un framework principalement front-end.

Remarque : *Sur les commandes angular.*
Ces commandes commencent par `ng` (dans un terminal).

Remarque : *Any desk.*
Un genre de team viewer (mais en ligne).

Remarque : *Pour créer un projet.*
`neg new NOM_PROJET`. puis (`n` : pas dans la dernière version), `y` (routing) et sélectionner un language de style.

Remarque : *Sur html et css.*
html sert à faire du formatage de page et css (fiche de style en cascade) à faire du formatage de style. Angular utilise les deux pour faire des pages web.

Définition : **SCSS**.
CSS avec une écriture moins lourde que css. Attention dans un navigateur on ne peut voir que du css. SCSS doit être compiler vers CSS avec Angular pour être interprété par un naviguateur.

Note :
Pour l'édition de code il faut ensuite passer par Visual Studio Code ou bien PhpStorm (payant mais un mois de licence gratuite...)

Remarque : *Pour excécuter des scripts (angular mais pas que).*
`set-executionpolicy unrestricted` dasn un powershell en mode administrateur (2 conditions obligatoires). Nécessaire pour travailler avec Angular.

Note :
Les projets Angular sont super Fat en terme de place sur le disque.

Note :

Il faut ouvrir le dossier de projet entier (dans VSCode) pour que le serveur puisse fonctionner par la suite.

Remarque : *Sur l'arborescence d'un projet.*

- `tsconfig.json` : config pour la compilation du Typescript vers du javascript (javascript est par contre un Typescript valide). Ce fichier est nécessairepour faire du typescript (et ce n'est pas propre à Angular).
- `package.json` : fichier de dépendance du projet. Si l'on utilise ce fichier la première chose à faire est un `npm install` : pour installer les dépendances d'un projet (va charger le package json dans le nodemodule).
- Dans le dossier `app` : `app.component.xxx` permet de partitionner l'application en component. Un compenent à un rôle précis → il gère un seule et unique action. Chaque componenet doit être l eplus indépendant possible. Exemple : le html gère le rendu d'affichage, le css les style, spec.ts sert aux test et le ts est le *controlleur* en lui même. Chaque nouveau component aura 4 fichier mon_component_component associés.
- Dans le dossier `app` : `app.module.ts` permet d'indiquer les librairies nécessaires à l'application (unqiuement ceux inclues dans l'application et pas tt les modules du node module).
- Dans le dossier `app` : `app-routing.modules.ts` : pour gérer les routes associées aux différents components. Ce fichier lie chaque component à un nom.
- Dans le dossier `index.html` : est la page principale du projet (dans laquel sera importer le projet après compilation).
- Dans le dossier `styles.scss` : fichier scss mère qui sera utilisé partout ailleurs dans le projet. Par contre le scss des components est généré en premier (ce qui est important car l'ordre de redéfinition des classe est important.).

Remarque : *En css.*

L'ordre des importation est très important. On peut modifier un import précédent avec un import suivant.

Définition : **Bootstrap**.

Une bibliothèque CSS qui fournit un ensemble d'objets.

Note :

Pour récupérer des données en Angular il faut forcément passer par une API, il n'est pas possible d'interroger directement une base de donnée.

Remarque : *Sur la compilation.*

Une fois la compilation et le serveur lancé (`np serve –open`) il n'est pas nécessaire de l'arréter. En JavaScript le code est rechargé en permanence.

Remarque : *Sur l'objet en Angular.*

En Angular (et typescript) tt est objet. On en revient donc au un fichier, une classe. Au niveau de la syntax d'une variable nom : type = valeur.

Mot clef : **let**.

permet de déclarer une varibale locale modifiable. Le typage n'est pas obligatoire mais est en théorie obligatoire.

Mot clef : **const**.
permet de déclarer une varibale locale non-modifiable. Le typage n'est pas obligatoire mais est en théorie obligatoire.

Mot clef : **any**.
à la place d'un type, permet de déclarer n'importe quel type. A oublier le plus vite possible (car l'intrêt du typescript est de typé au maximum le code.)

Note :
A partir du moment ou un attribut ou une fonction (dans le component.ts) est public (visibilité par défaut) il est possible de l'utiliser dans la partie html du component.

Mot clef : **export**.
devant la déclaration d'une classe. permet de dire que la classe va pouvoir etre utilisée en dehors du fichier de déclaration. Sans `export` le componet ne peut être utilisé nulle part.

Remarque : *ng-template.*
est un template qui est masqué sur une page tant qu'il n'a pas été appelé. Peut servir avec un *ngIf. Voir Cours et exemple

Mot clef : **Doc html/css**.
regarder ce site en cas de questions sur les languages balises à la con.

Remarque : *Pour créer un nouveau component.*
ng generate component component_name (ou ng g c component_name si on passe par les alias).

Mot clef : **implements**.
permet d'implémenter une interface.

Mot clef : **extends**.
permet de déclarer un héritage.

Remarque : *Sur l'interface OnInit.*
fait partie du cycle de vie des objets de type component. OnInit indique un traitement appliqués après l'initialisation d'un objet (constructeur, injection de dépendance). Le OnInit est fait avant que la vue soit générer. En générale on fait dans le OnInit les requêtes à l'api, gérer les paramètres de root.

Sur le cycle de vie des objets angular.

Remarque : *Sur OnDestroy.*
Est appelé lorsque la fenêtre associée au component disparait.

Remarque : *pour installer une dépendance/librairie.*
`npm install un_truc` ou `npm i un_truc` si l'on passe par les alias. Met à jour le package.json.

Remarque : *Sur le SCSS.*
On peut définir des variables en SCSS en préfixant le nom d'un d'un dollar : $variable . & rappel l'élément courant (parent plutôt, un peu l'équivalent du this).

Remarque : *console.log(string).*

permet d'afficher une string dans la console log. Si on lui passe un objet la console affichera tt les attributs de la console

Remarque : *[ngStyle]="{'x'; source}"*.

est une directive Angular qui permet de chercher une propriété dans une classe source pour l'affecter ici un Style html. Permet de binder à la propriété Style la valeur x extraite de source. Pour plus de détails voir : https://angular.io/api/common/NgStyle.

Mot clef : **super**.

permet de rappeler le constructeur de la classe mère. Dans le constructeur c'est la première instruction qu'il faut appeler. En dehors on peut l'appler à tt moment.

Remarque : *rem*.

unité de mesure qui se base sur la police du body (environnement le plus extérieur dans une page). Pour plus de détail sur les tailles : https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units

Remarque : *Sur le padding*.

Il s'agit de l'écart entre entre le placement d'objet dans unvironnement et les bords de l'environnement. Il faut regarder sur une page (dans les outils de debug) pour visualiser sa.

Remarque : *sur le *ngFor*.

Ce dernier peut fonctionner sur des container entier tant qu'ils sont contenue à l'intérieur de ces derniers.

Remarque : *Sur le dossier `asset`*.

Dans ce dossier ; tt les ressources nécessaires pour le site (images, musique, video, gif...).

Remarque : *Pour afficher une image*.

balise <img src = "chemin">. Attention la balise ne se ferme pas comme les autres. On peut également utiliser le Binding d'Angular [img]=hero.image au lieu de src = "{{hero.image}}".

Remarque : *Pour régler les problèmes de taille d'image*.

Il faut faire un environnement ou container-image qui est définit dans le .scss avec un enfant de type balise image (déclaré > img{width = x%, height = y%} dans le scss. Se lit pour tt balises image contenues dans un container-image alors on appliques ces règles).

Remarque : *sur [class.une_classe] = condition*.

Signifie que l'on applique la classe css si la condition est vrai.