**Laboratoire #1** (Partie 1 sur 2)

Je réalise mon premier projet Angular 👶

**Étape 1 - Créer le projet Angular 🥳**

* **11**  Créez un projet **Angular✨** à l’aide de Powershell.
  + Votre projet doit être nommé **lab1**
  + On ne souhaite pas utiliser de routage.
  + Le format pour les feuilles de styles est CSS. 🎨
* **12**  Installez les dépendances nommées « **bootstrap** » et « **jquery** » (N’oubliez pas de vous déplacer à L’INTÉRIEUR du dossier de votre projet avant de taper la commande ! Attention aux fautes de frappe !)
  + Vérifiez qu’une ligne représente bel et bien chaque dépendance dans **package.json**



* + Ouvrez votre projet avec **Visual Studio Code**.
  + Fermez Powershell, car ça fait plus propre.

**Étape 2 - Bootstrap ? Non, jamais entendu parler 🙄**

* **28**  Tentez d’exécuter votre application Web une première fois pour regarder la page placeholder initiale.
  + Videz le template HTML du composant « **app** », car vous allez le remplir vous-mêmes.
  + Supprimez le fichier de test du composant **app**. On n’en aura pas besoin. 
* Nous devons faire une petite manipulation pour que les classes **Bootstrap** fonctionnent dans notre projet.
  + Dans le fichier **angular.json**, trouvez les propriétés "styles" et "scripts" (**Attention !** Ces deux propriétés sont présentes chacune 2 fois. Ce sont les premières qu’on doit remplir) et écrivez ceci :

"styles": [

"./node\_modules/bootstrap/dist/css/bootstrap.css",

"src/styles.css"

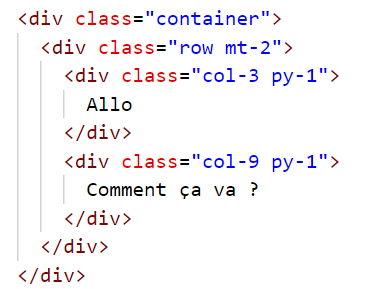
],

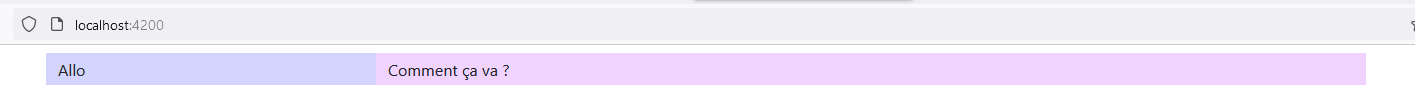
"scripts": [

"node\_modules/jquery/dist/jquery.min.js",

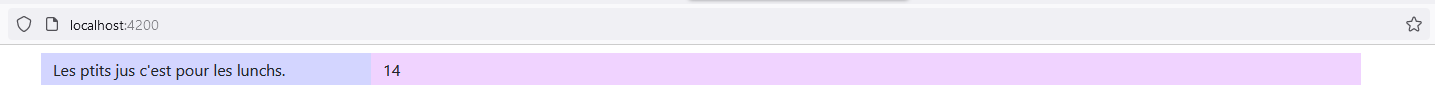
"node\_modules/bootstrap/dist/js/bootstrap.min.js"

]

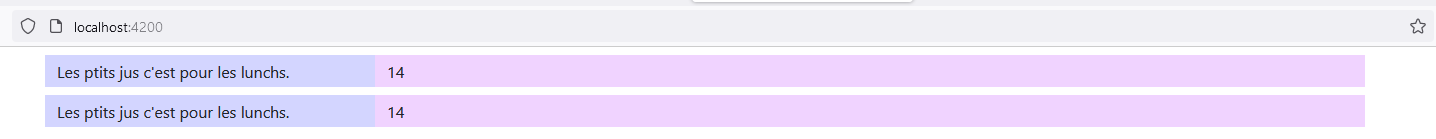
* Revérifiez qu’il n’y a pas de faute de frappe. (Quoique ce serait spécial si vous avez copié-collé)
* Redémarrer votre application Web pour que ce changement soit effectif : Dans le terminal de **Visual Studio Code**, mitraillez **Ctrl + C** pour éteindre l’application Web. Essayez de réactualiser votre page localhost:4200 et si elle ne veut plus charger, c’est bon, l’app est fermée. Vous pourrez alors la relancer avec **ng serve**.
* **29-32**  Modifiez le template HTML du composant « **app** » pour avoir un résultat similaire à celui ci-dessous.
  + Vous devez utiliser des classes Bootstrap et du CSS pour arriver à ce résultat.
  + Pour modifier le CSS, il faut toucher au fichier **app.component.css**. (S’il y consent)
  + La première colonne prend environ 25% de la largeur du container.
  + Vous pouvez choisir d’autres couleurs de fond car vous êtes très créatifs. 🎨



* **33-34**  Créez une variable de classe nommée « **myWisdom** » dans le composant « **app** » et initialisez immédiatement sa valeur avec une phrase qui inspire votre quotidien.
  + Affichez la valeur de cette variable dans la colonne de gauche de votre template HTML **app**.
  + Supprimez la variable « **title** » qui était déjà présente de base.
* **33-34**  Créez une DEUXIÈME (wow !) variable de classe nommée « **n** ». Typez cette variable avec le type **number**. Ajoutez un **constructeur** sans paramètre qui définit la valeur de **n** avec un nombre de votre en choix entre 1 et 13. (Mais pas plus que 13 !! 😠)
  + Affichez la valeur de cette variable dans la colonne de droite.

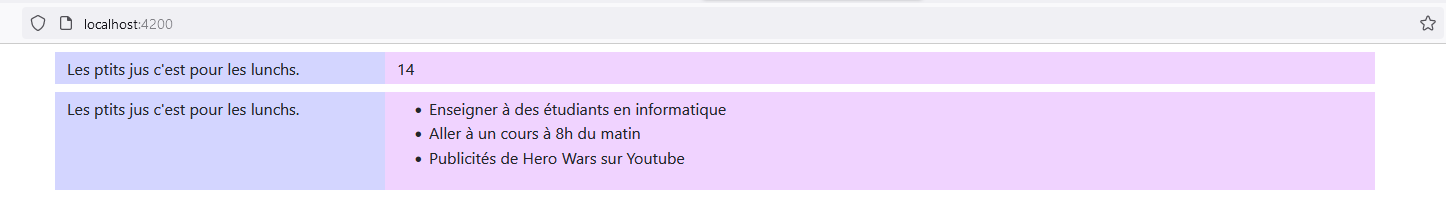


* Copiez-collez votre .row et son contenu dans le HTML :



**Étape 3 : Liste et pitous 🐕**

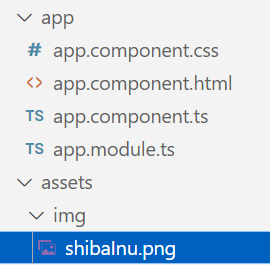
* **35-38**  Nous allons afficher une **liste** de string avec trois à cinq éléments dans la 2e rangée, dans la colonne de droite.
  + Dans votre composant « **app** », ajoutez une variable de classe nommée **hateList**.
    - Elle doit être initialisée immédiatement avec **3 à 5 chaînes de caractères** qui représentent des choses que vous détestez.
  + Dans le template HTML, en utilisant **\*ngFor**, **<ul>** et **<li>**, affichez tous les éléments de votre hateList à l’endroit demandé.



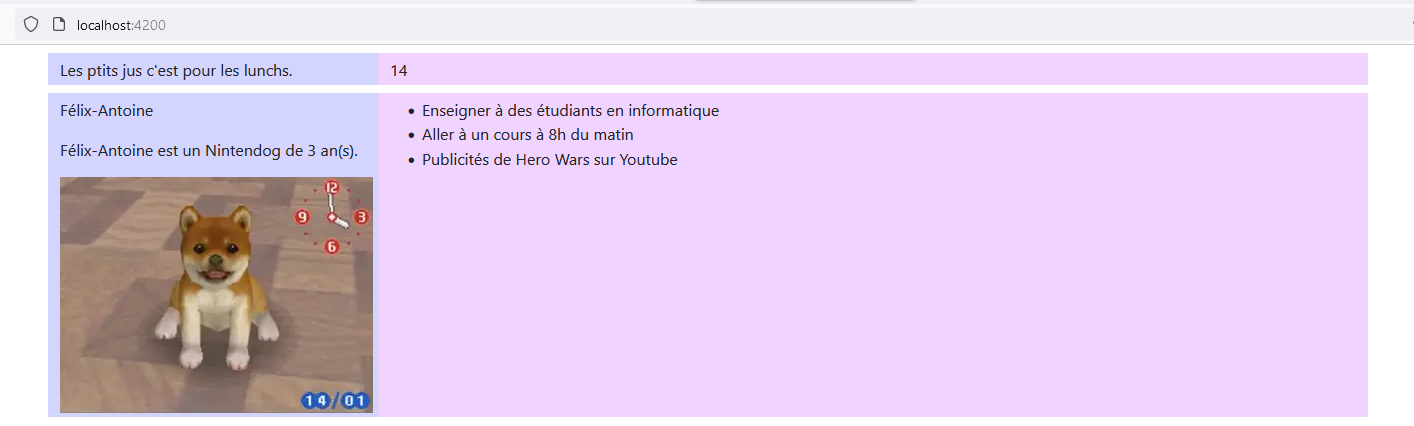
* **18-23, 33-38**  Nous allons afficher les propriétés d’un **objet** ainsi que le résultat d’une *méthode* dans la 2e rangée, dans la colonne de gauche.
  + Créez une nouvelle **classe** sous la classe **AppComponent**. Elle doit se nommer « Nintendog».
    - Cette classe possède une propriété 🧪 :
      * **age** : Un *number* avec l’âge du Nintendog, immédiatement initialisé avec un nombre de votre choix entre 0 et 10.
    - Cette classe possède un constructeur 🧰 :
      * Il reçoit un paramètre : **name**, qui est un *string*. Ajoutez le petit **mot-clé** devant ce paramètre pour en faire automatiquement une variable de classe.
      * Il reçoit un autre paramètre : imgUrl, qui est un string. Ajoutez encore le petit **mot-clé** devant.
      * Le corps du constructeur est vide.
    - Cette classe possède une méthode ⚙ :
      * **dogInfo()** : Aucun paramètre. Retourne ce string :

**this.name + " est un Nintendog de " + this.age + " an(s)."**

* + Sur *Gougueule Images*, trouvez une image de **chien Nintendog** et placez-la ici :



* + Dans la classe de votre composant « **app** », ajoutez une nouvelle variable nommée « **dog** » et initialisez-la avec une donnée de type **Nintendog** créée à l’aide du constructeur de la classe.
    - Pour la propriété imgUrl, la valeur sera **"/assets/img/monImage.png"**.
  + Dans le template HTML, affichez, dans la 2e rangée, à gauche, trois informations sur votre Nintendog :
    - Son nom dans un **<p>**.
    - Le résultat (texte retourné) de la fonction **dogInfo()** de votre chien dans un **<p>**. (Vous pouvez faire **{{monObjet.saMéthode()}}**
    - L’image de votre chien avec un **<img>**. N’hésitez pas à lui donner la classe **"img-fluid"** si elle est trop grande. L’attribut **src** peut contenir "**{{monObjet.sonLienImg}}"**



* Bon allez, c’est assez pour aujourd’hui. Allez jouer dehors un peu. ☃ Sauvegarder votre projet avec Git ou bien sur une clé USB / dossier OneDrive.
  + Si vous choisissez une clé USB / OneDrive, supprimez les dossiers **node\_modules** et **.angular** avant ! 😠

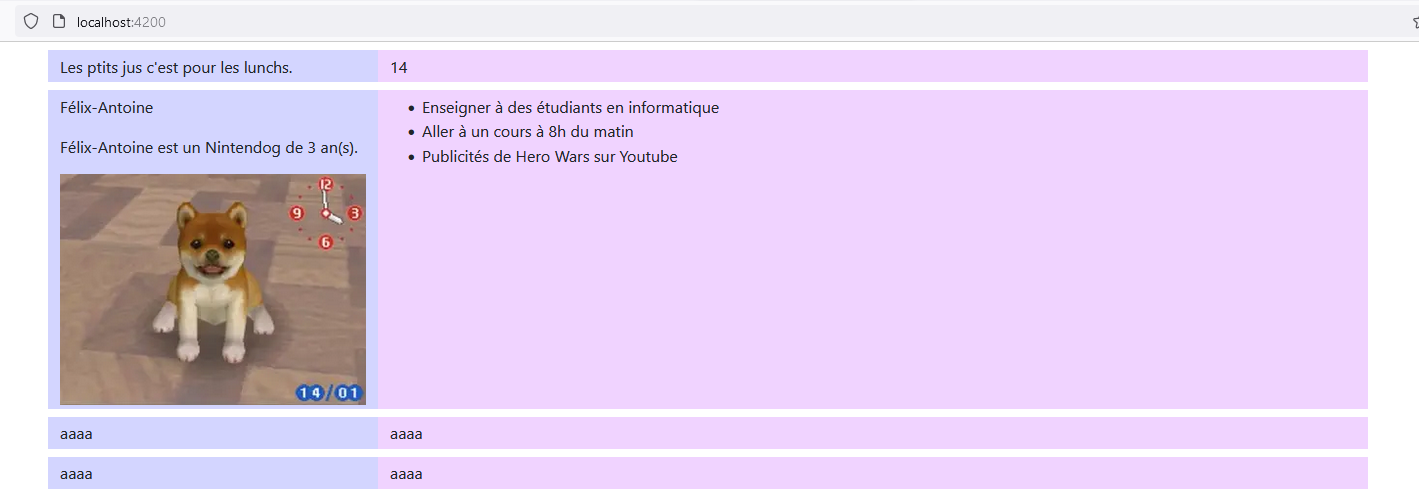
**Laboratoire #1** (Partie 2 sur 2)

**Étape 4 - Qui a vidé mon dossier node\_modules ? 😠**

* Récupérez le projet Angular tel que vous l’aviez stocké à la fin de la partie 1 sur 2.
  + Normalement, le dossier **node\_modules** avait été vidé pour alléger le stockage. Si ce n’est pas le cas, remontez le temps et faites la partie 1 comme il faut. ⏳
  + **14**  Toutefois, pour que notre application puisse être exécutée, vous êtes invités à réinstaller toutes les dépendances à l’aide de la bonne commande.
  + Vous pourrez ensuite ouvrir votre projet dans Visual Studio Code.
  + Exécutez votre projet avec ng serve pour vous assurer que tout fonctionne bien.
    - Si oui, vous pourrez commencer à faire les prochaines modifications.

**Étape 5 - Ajustements dans le HTML** 📐

* Nous allons faire quelques ajustements à notre template HTML pour les prochains exercices.
  + Ajoutez simplement 2 **.row** à votre **.container**. Elles peuvent être identiques à celles déjà présentes.

****

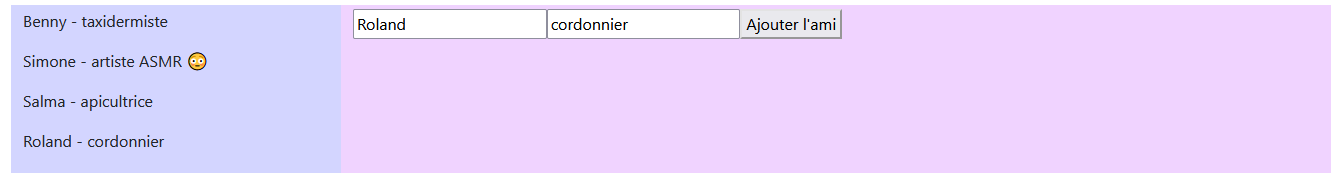
* **38**  Sauf si c’est déjà fait, déplacez votre classe Nintendog dans son propre fichier.

**Étape 6 : Formulaire et amis imaginaires** 📝

* Pour cette étape, nous allons créer une classe **ImaginaryFriend**, afficher une liste d’amis imaginaires dans la page et créer un **formulaire** qui permet d’ajouter des amis imaginaires supplémentaires dans la liste affichée.
* Créez une classe nommée **ImaginaryFriend**, dans son propre fichier :
  + Propriété(s) 🧪 :
    - Aucune qui soit explicite.
  + Constructeur 🧰, qui prend 2 paramètres avec le mot-clé **public** devant :
    - **name** : Un *string*.
    - **occupation** : Un *string*.
    - Le corps du constructeur est vide.
* Dans la classe du composant « **app** », ajoutez la variable **friends** et initialisez-la immédiatement avec une liste qui contient 3 **amis**. Assurez-vous qu’au moins un des trois **amis** est un "artiste ASMR".
* **31-33**  Dans la colonne de droite de la dernière rangée, affichez votre liste d’amis. (Avec **\*ngFor**, obligatoirement) Chaque objet de type **ImaginaryFriend** doit être dans une balise **<p>** séparée.
  + Au sein d’une balise **<p>**, on affiche le name, puis l’occupation. Si l’occupation est "artiste ASMR", on ajoute <span>😳</span> après l’occupation. **Astuce** : on ne peut pas utiliser facilement des guillemets dans un attribut HTML comme **\*ngIf**. À la place, utilisez des apostrophes pour encadrer une chaîne de caractères comme **'artiste ASMR'**.



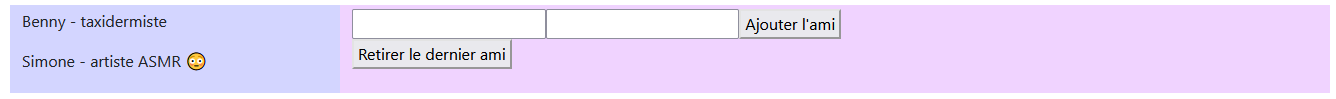
* **43-47**  Parfait, on a une liste et on l’a affichée. Maintenant on va créer le formulaire qui permet d’ajouter des **amis** à cette liste. (On le placera dans la colonne de droite de la même rangée.
  + Créez deux **input**s de type text. Faites un *2-way-binding* avec deux variables nommées **name** et **occupation**. (Variables de classe pour votre composant « **App**»)
    - Ces variables doivent pouvoir être égale à **undefined** initialement, car les champs <input> seront vides.
    - N’oubliez pas qu’il faut importer FormsModule.
  + Ajoutez un bouton pour **submit** avec le texte « Ajouter un ami ».
  + Le formulaire doit appeler une fonction nommée **addFriend()** lorsqu’il est **submit**.
  + Codez la fonction **addFriend()** dans votre classe **AppComponent**.
    - Cette fonction utilise les variables **name** et **occupation**, en vérifiant qu’elles ne sont pas *undefined*, pour instancier un nouvel objet de type **ImaginaryFriend** et l’ajouter à la liste **friends**. Comme ceci : **this.friends.push(new ImaginaryFriend(…))**
  + Vérifiez que votre formulaire fonctionne bien et que l’affichage de la liste est dynamique.



* + Petite amélioration : à la fin de la fonction **addFriend()**, dans le if, ajoutez ce code pour que les input se vident :
    - **this.name = "";**
    - **this.occupation = "";**

**Étape 7 - Un petit événement**

* **40-42**  Ajoutez un nouveau **<button>** juste en-dessous du **<form>**. (À l’extérieur du **<form>**)
  + Sur le bouton, il doit être écrit « Retirer le dernier ami ».
  + Cliquer sur le bouton doit appeler la fonction « **removeFriend()** ».
* Ajoutez une fonction nommée « **removeFriend()** » dans votre composant « **App** ».
  + Cette fonction retire le dernier élément de la liste **friends**. En JavaScript, on peut faire cela facilement grâce à la méthode this.maListe.**pop()**.
* Vérifiez que votre nouveau bouton fonctionne et que l’affichage de la liste se met à jour.



**Étape 8 - Promis c’est la dernière étape**

* Copiez-collez ce CSS dans **app.component.css**

.dark{

    color:white;

}

.dark .col-3{

    background-color:#161958

}

.dark .col-9{

    background-color:#3b1149

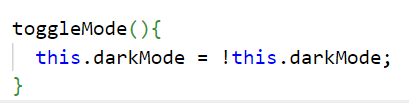
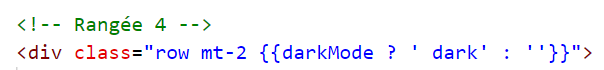
}

* Faites-le nécessaire pour rendre un bouton fonctionnel dans la dernière rangée qui permet d’alterner entre un thème clair et un thème obscur. (Pour mettre un émoji dans le HTML appuyez sur **Windows + ;** )





* Voici trois indices :



**Étape 9 - Buh-bye**

* Sauvegardez votre laboratoire quelque part.
* Les remises ne sont jamais obligatoires pour les laboratoires, mais vous pouvez utiliser les remises sur Léa comme point de sauvegarde.
* N’oubliez pas de supprimer **node\_modules** et **.angular**, à moins d’utiliser Git.

~|~  Il reste juste 14 semaines wow ~|~