**Etape 0 :** installer nvm (node version manager) <https://github.com/coreybutler/nvm-windows/releases>

**Etape 1 :** installer node.js (environnement d’exécution js) <https://nodejs.org/fr>

 permet d’instaler node.js et npm



**Etape 2 :** vérifier l’installation de npm (permets d’installer des packages js)



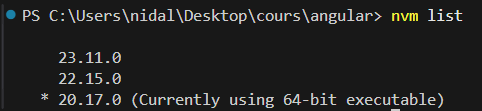
**Exécuter un fichier js dans la console :**

1. créer un fichier hello.js et écrire dedans 

2. Exécuter 

**Etape 3 :** installer angular

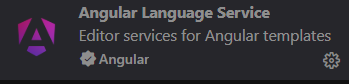
 ne pas utiliser la dernière version de node, mais la 20 plutôt

Permet de vérifier la version utilisé par angular

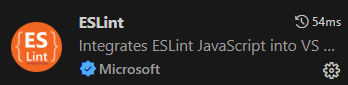
 Installer angular

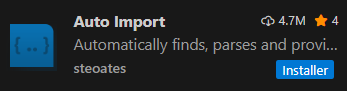
 permet de vérifier la version d’angular

**Extension VS code à installer :**









**Créer un nouveau projet angular**

 Créé un nouveau projet nommé « playing-cards » dans le répertoire courant du terminal

Voici les réponses que l’on donne à la suite de la création d’un nouveau projet

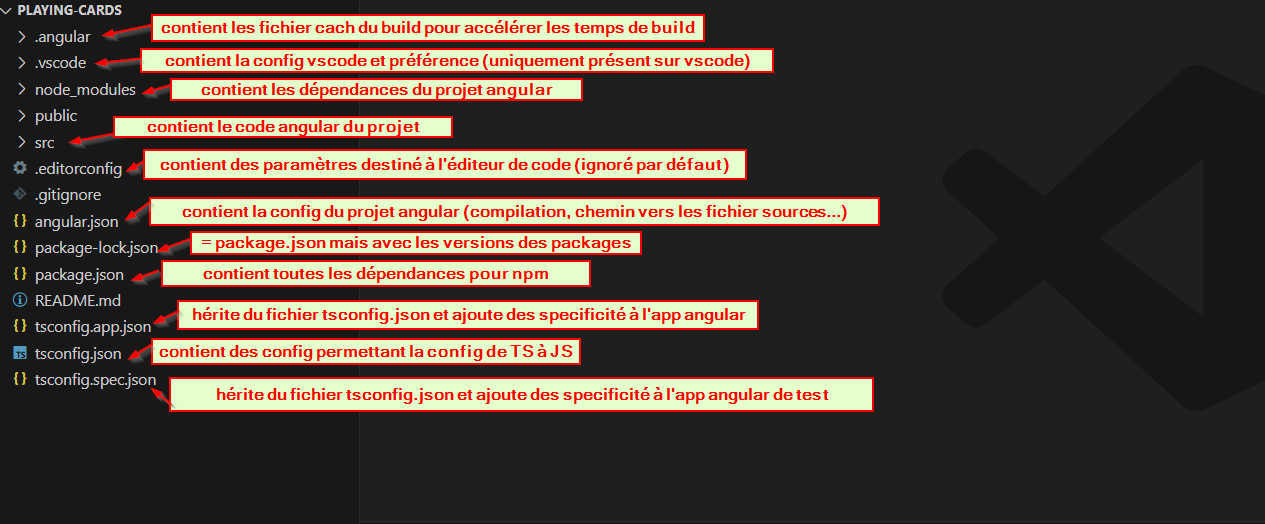


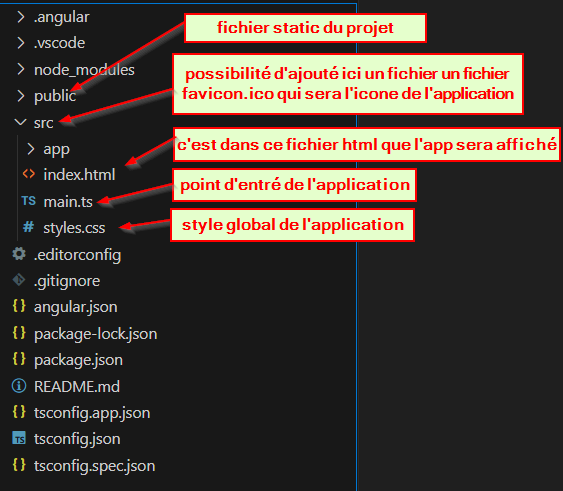
**Lancer le serveur du projet**

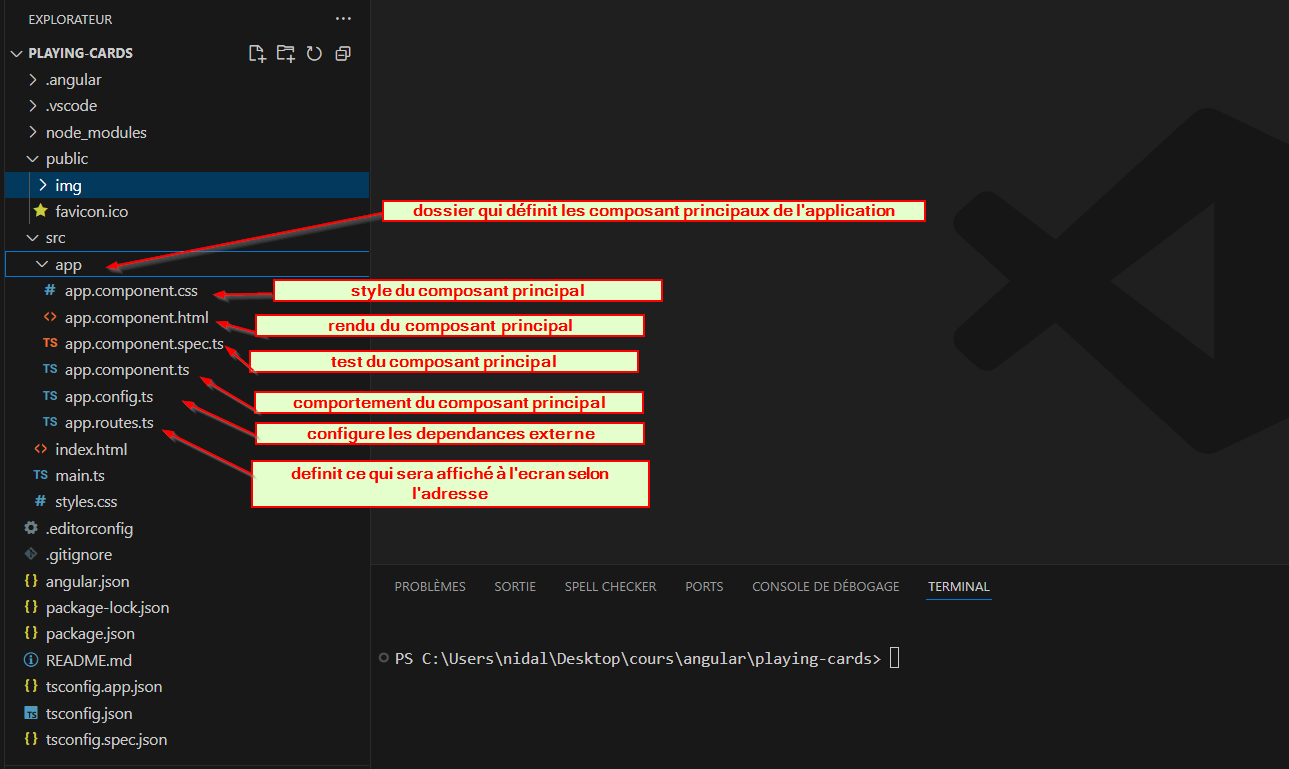
Ouvrir le dossier du projet dans un IDE puis



**Description des fichiers du projet**

****

****

****

**Comment accéder au fichier static du projet stocké dans le dossier public**

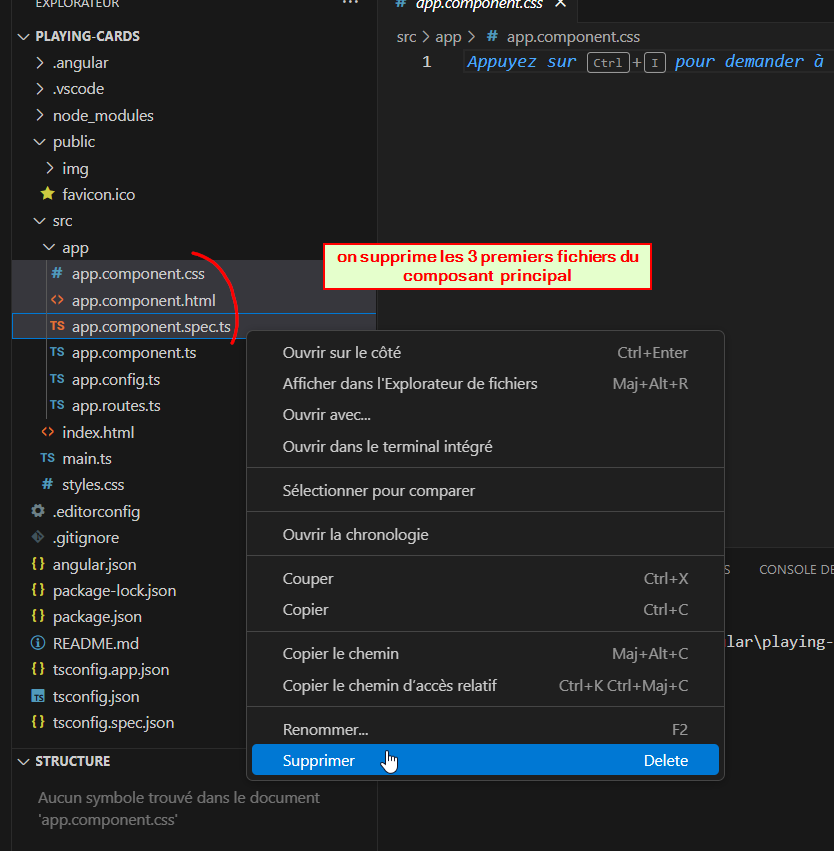
****

**Un composant :**

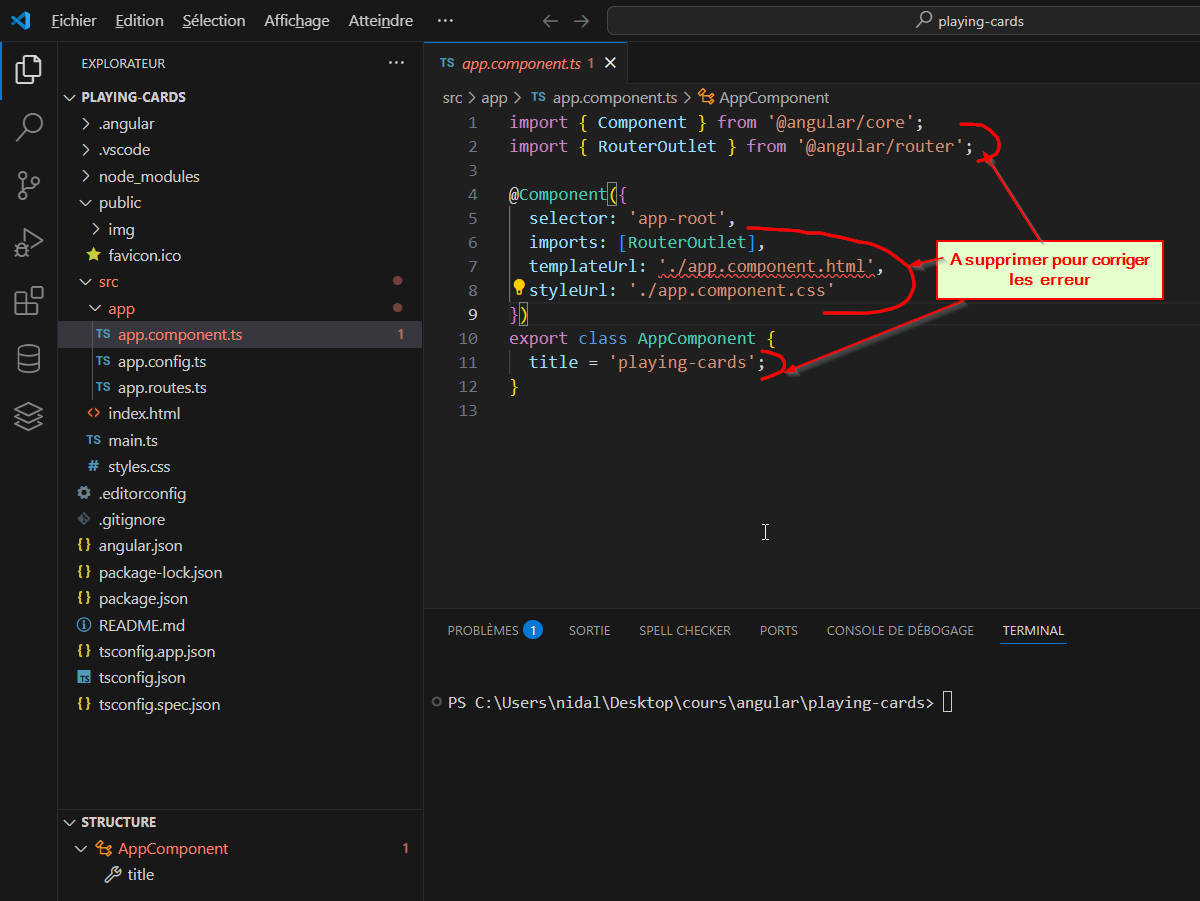
C’est une brique de notre application qui s’occuper d’un aspect bien spécifique de notre projet et contient:

* Le style (.css)
* Le rendu (.html)
* Le comportement (component.ts)
* Les test (spec.ts)

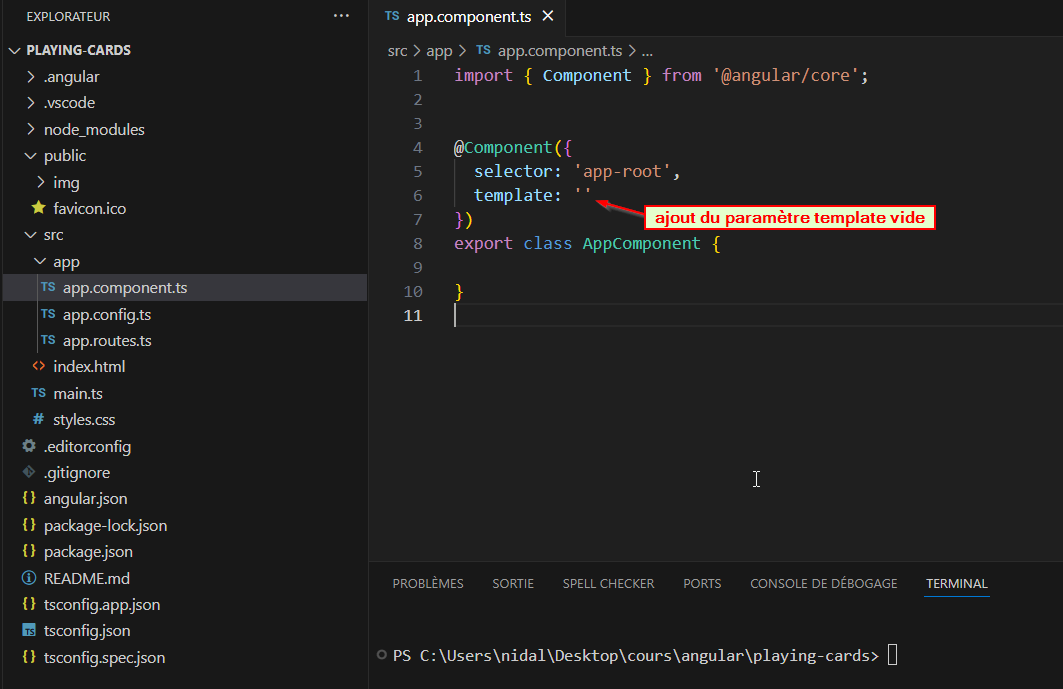
**On supprime les 3 premiers fichier du composant principal**

****

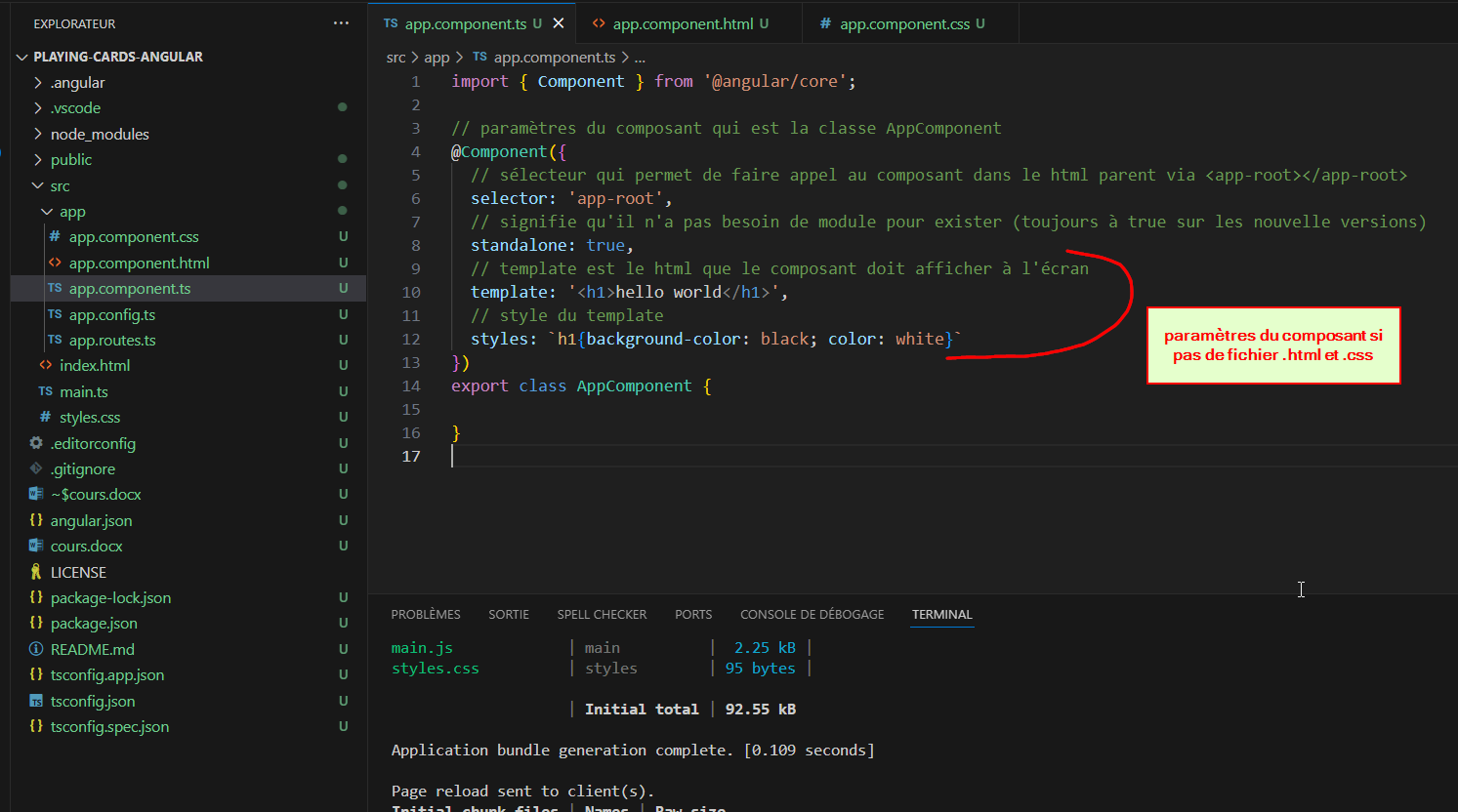
**Après suppression des 3 fichiers du composant principal, se rendre dans app.component.ts et supprimer ceci pour corriger les erreurs**

****

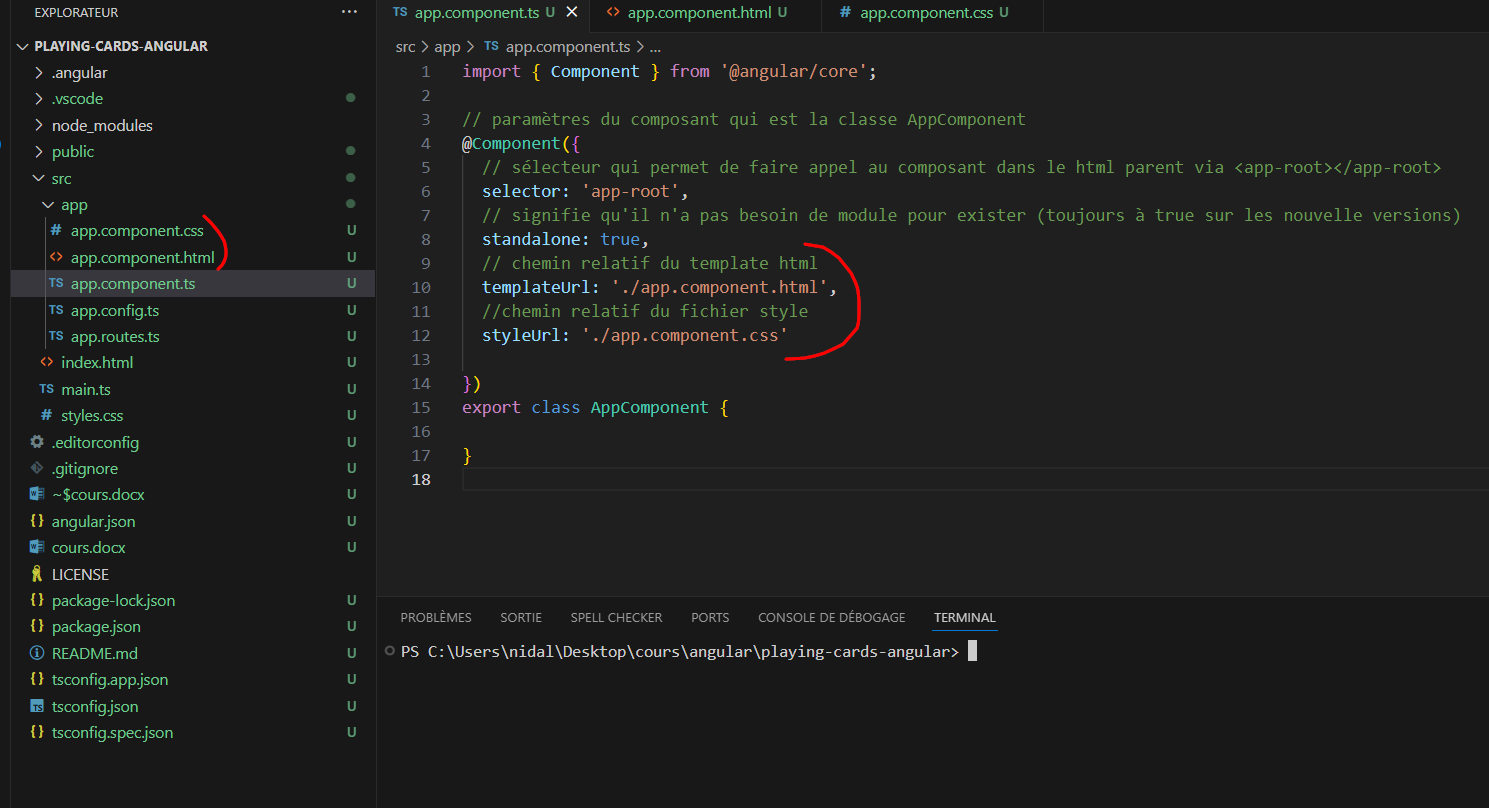
**Puis ajout du paramètre template vide au @Component**

****

**Voici les paramètres du fichier app.component.ts si pas de fichier .html et .css**

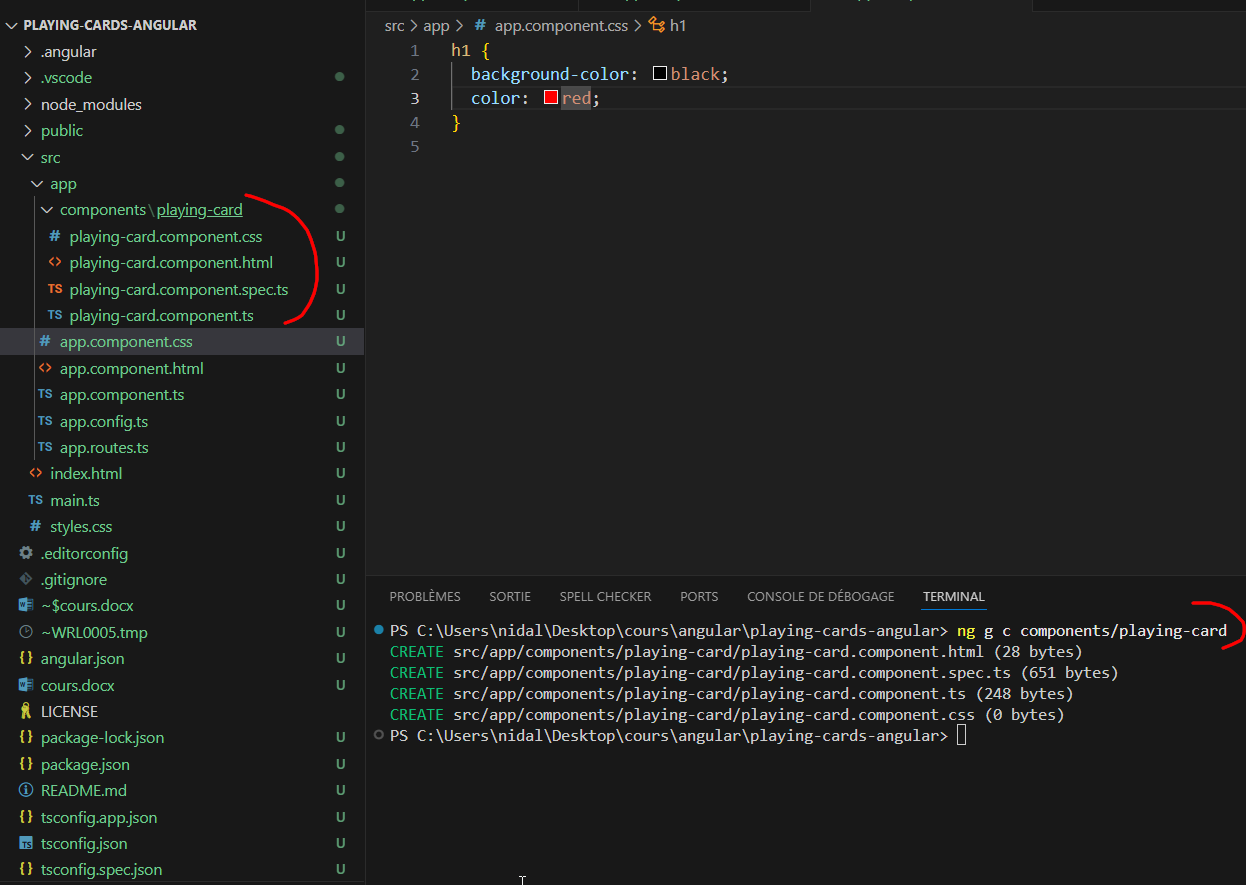
****

**Si on ajoute dans le dossier app/ les fichier app.component.css et app.component.html voici les paramètres à utiliser**

****

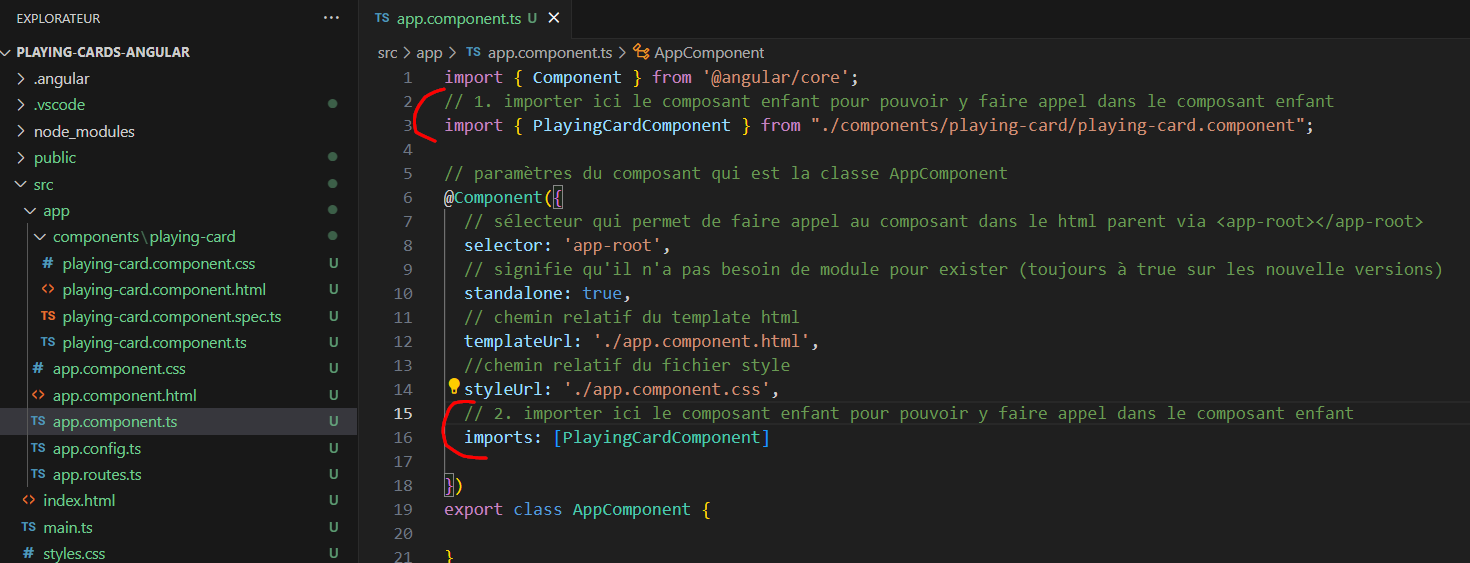
**Créer un nouveau composant**

**** Commande raccourci permettant de créer un nouveau composant nommé « playing-card » se situant dans le dossier components

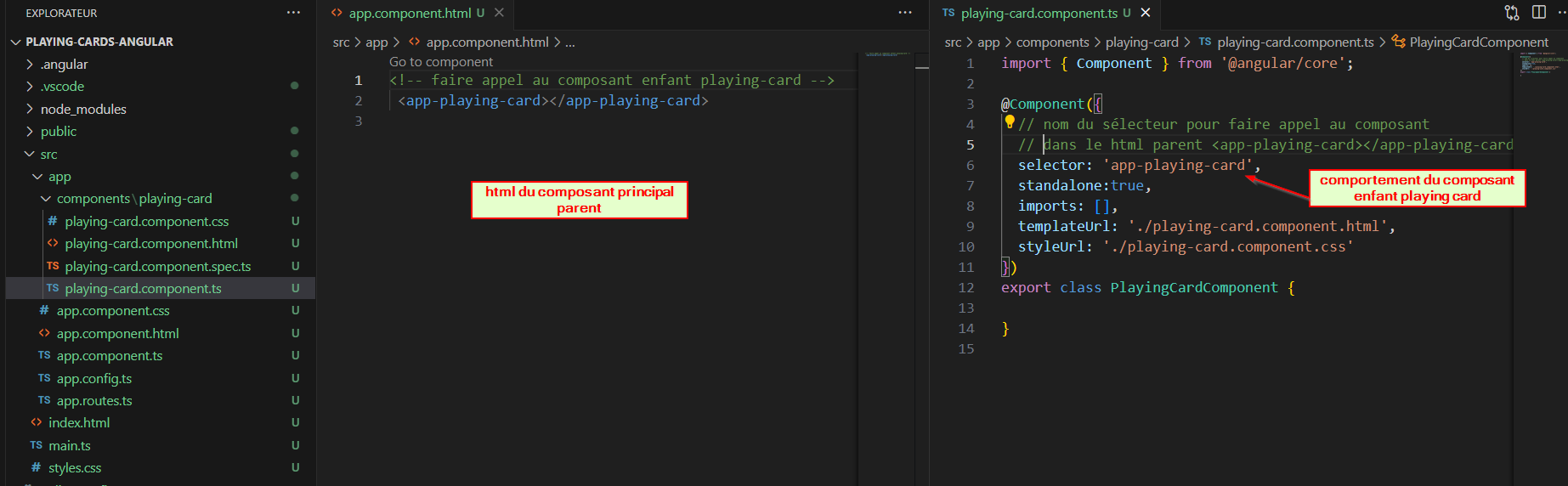
****

**Faire appel à au composant enfant playing-card dans le composant principal parent**

**(Se fait automatiquement) mais d’abord importer le composant enfant dans le composant parent principal app.component.ts**

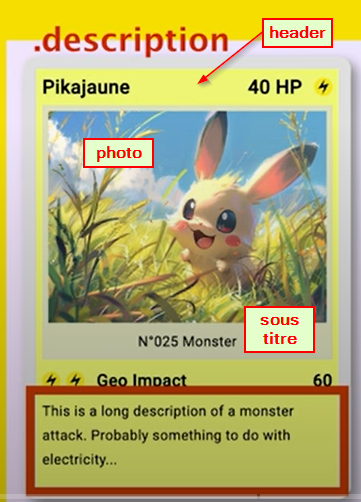
****

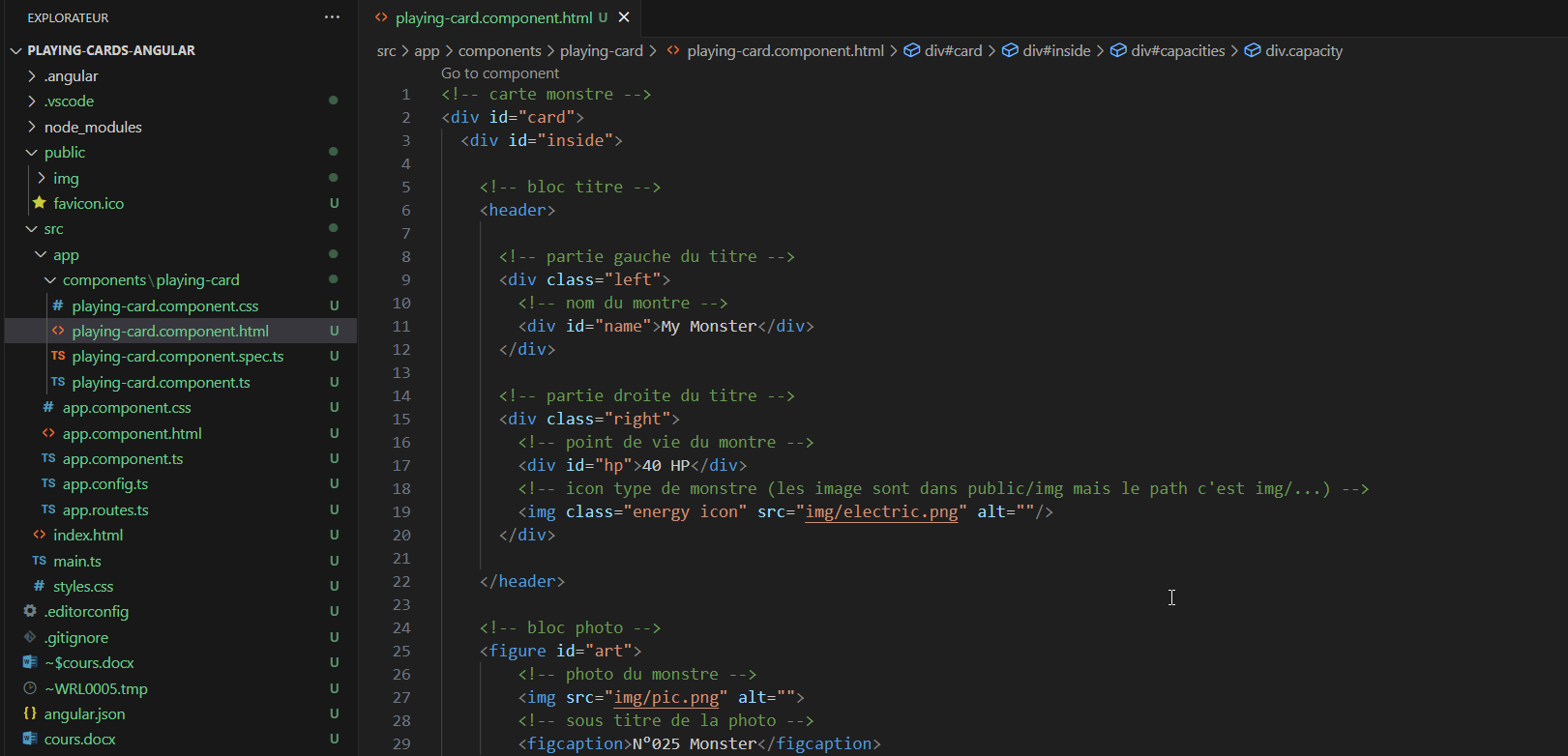
**Puis utiliser la baliser html du sélecteur du composant enfant**

****

**Modifier playing-card.component.html pour y insérer une card**

Tout d’abord créer la partie html : ici nous avons mis en place un template de carte pokemon

****

****

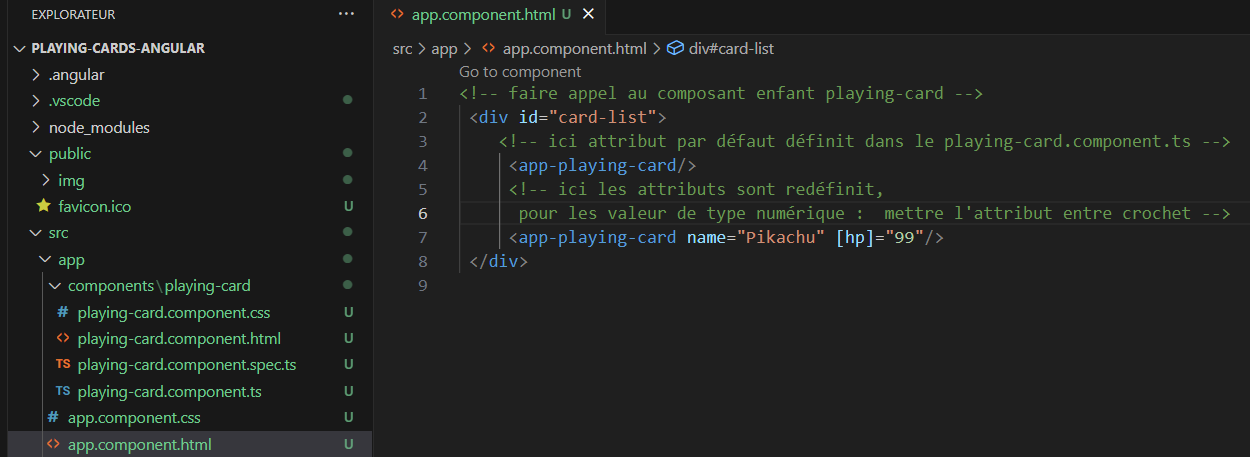
**Puis construire le style de la card html dans playing-card.component.css**

**Puis nous avons la possibilité de modifer le fichier style.css qui a étendu global (font-family, background de la page)**

**On initialiser les attributs d’un monstre dans la classe PlayingCardComponent du module playing-card.component.ts**

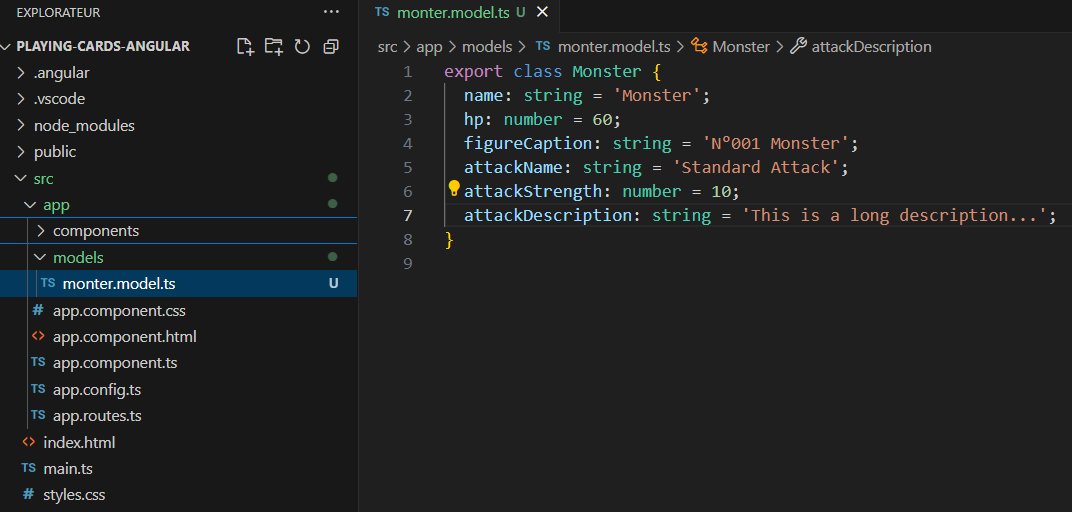
****

**Pour utiliser les propriétés qu’il y a dans le playing-card.component.ts on fait appel aux attribut de la classe PlayingCardComponent entre accolade dans le html du composant principal parent tout en ayant la possibilité de les redéfinir avec les Input() html**

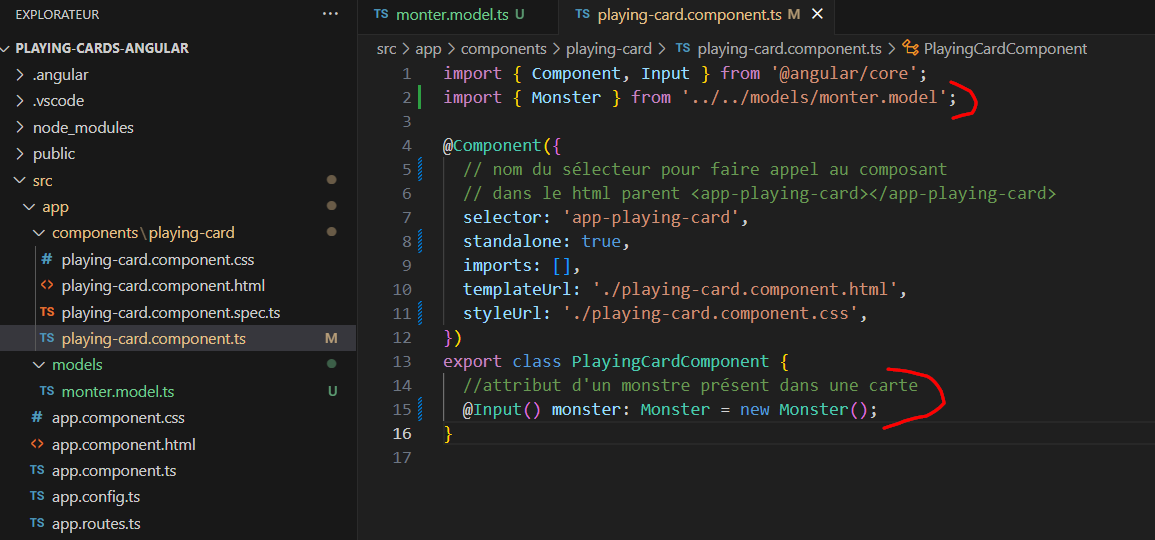
****

**Pour créer un nouvel model Monster**

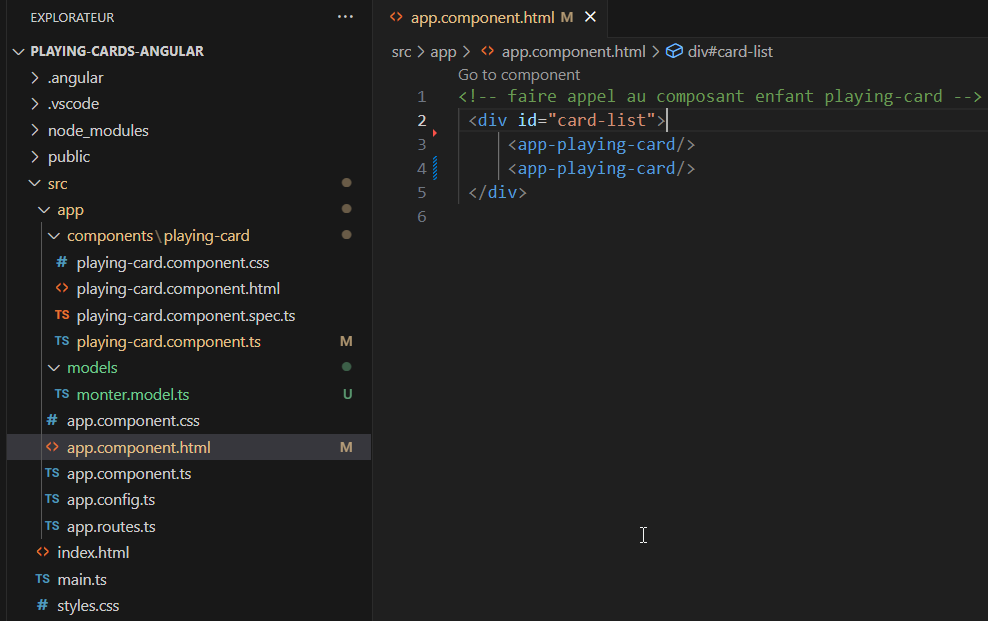
**Créer un nouveau dossier nommé « models » dans lequel on va créer un module monster.model.ts**

****

**Puis dans playing\_card.component.ts on instancie un Monster au lieu d’afficher des attributs qui ne sont pas en rapport avec la class PlayingCardComponent**

****

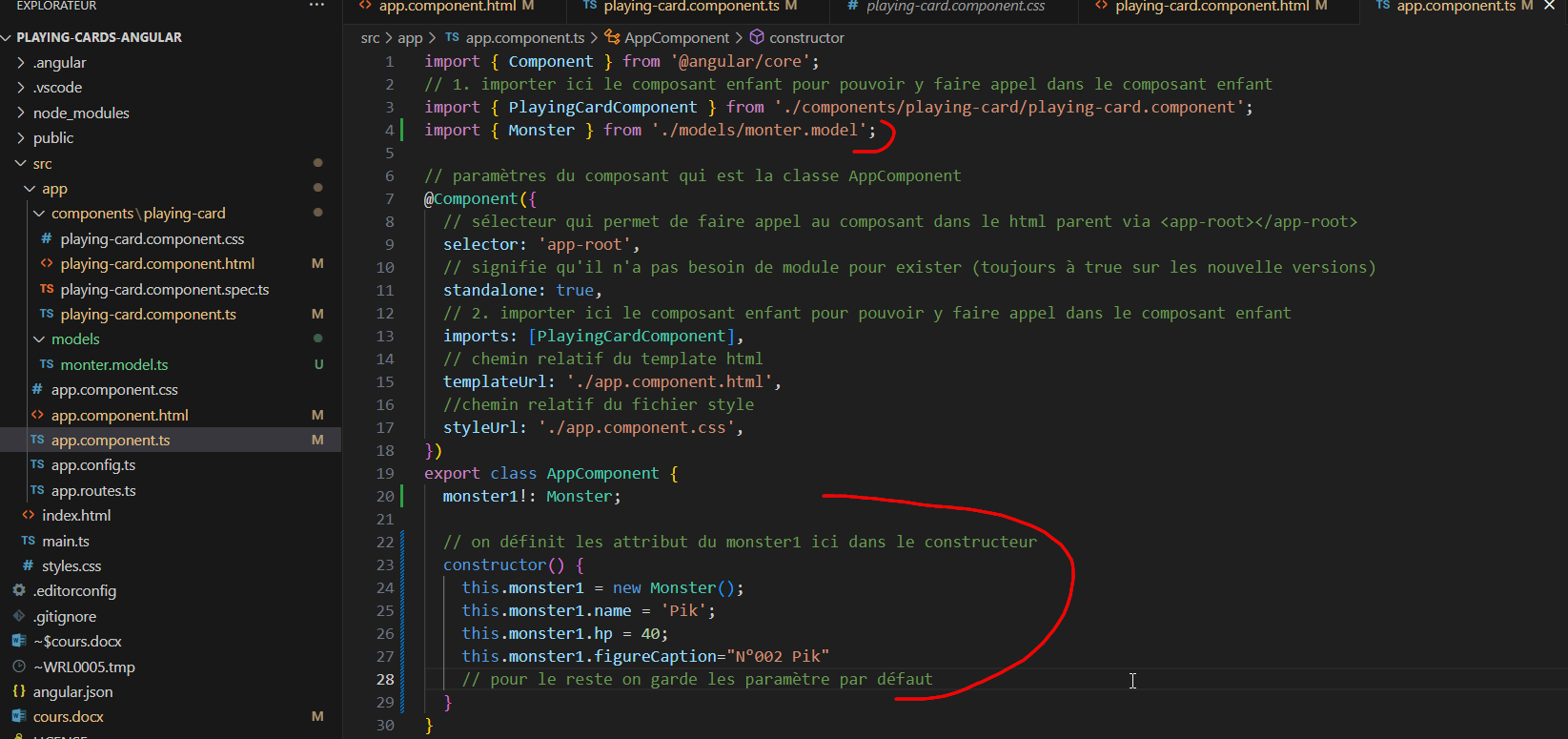
**Il est nécessaire de supprimer les input html dans le app.component.hmtl car les attribut ne sont plus disponible**

****

**Puis enfin supprimer les attributs de playing-card.component.html qui ne sont plus disponible**

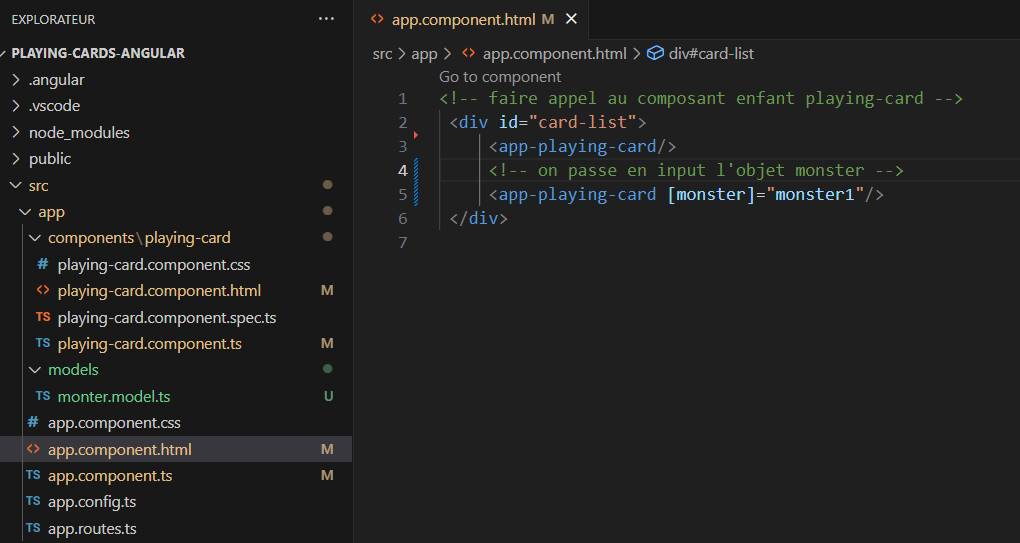
****

**Dans app.component.ts on va initialiser et définir un nouveau monster que l’on pourra passer en paramètre du app.component.html**

****

Le « ! » signifie que l’indique à ts que l’on va initialiser monster1 plus tard dans le code : dans le constructor

**Dans app.component.html on passe en paramètre monster1**

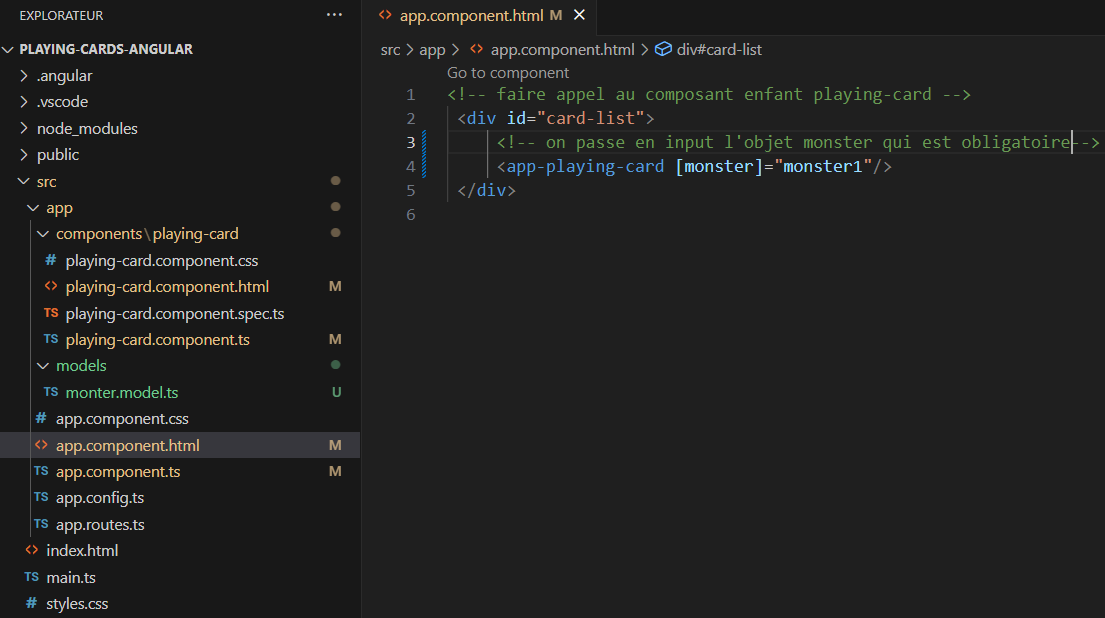
****

**On a aussi la possibilité de définir que l’argument « monster » est obligatoire dans le html parent app.component.html**

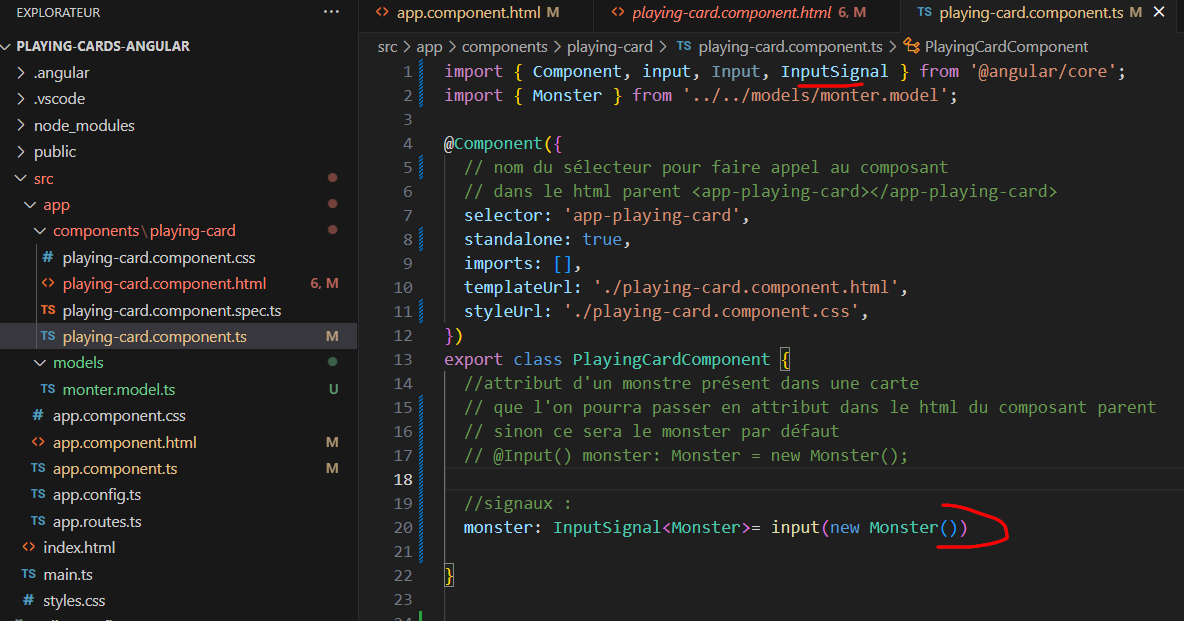
* **Se rendre dans playing-card.component.ts et ajouter le paramètre « required :true » dans le Input()**

****

**On supprime dons les balise html dans app.component.html qui n’ont pas de paramètre monster définit car cela provoque maintenant une erreur**

****

**Les signaux**

****

**Si on a un signal dans le component, dans le html du component courant, il faut ajouter des parenthèses après le nom de l’objet**

****