FORMATION ANGULAR 14

Angular est un **framework JavaScript** qui vous permet de développer des applications "efficaces et sophistiquées", comme dit la documentation. Il permet notamment de créer ce qu'on appelle des **Single Page Applications** (ou **SPA**) : des applications entières qui tournent dans une seule page HTML grâce au JavaScript.

Un **framework** logiciel est un ensemble d'outils et de composants à partir desquels on peut développer des applications.

Le développement Angular passe par trois langages principaux :

- le **HTML** pour structurer – toutes vos connaissances avec ce langage vous seront utiles, et Angular viendra vous ajouter quelques nouveautés ;
- le **SCSS** pour les styles – le SCSS est une surcouche du CSS qui y apporte des fonctionnalités supplémentaires, mais qui permet également d'écrire du CSS pur si on le souhaite ;
- le **TypeScript** pour tout ce qui est dynamique, comportement et données – un peu comme le JavaScript sur un site sans framework.

TypeScript

Nous verrons le typescript ensemble avec Angular. Don't Panik !

Documentation Officielle de typescript

Installation environnement

Node js LTS

Angular@latest

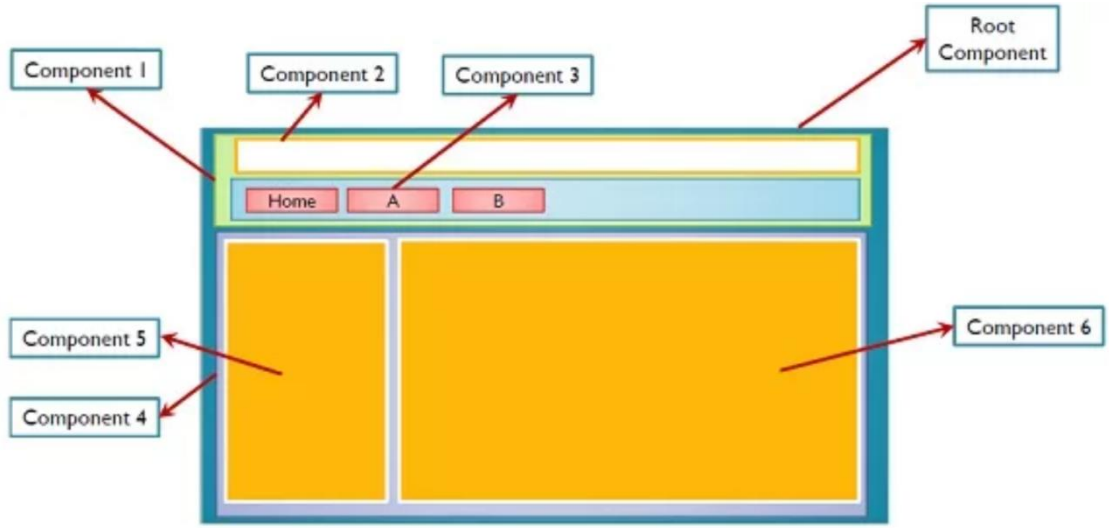
plugins VsCode

⇨ https://marketplace.visualstudio.com/items?itemName=Angular.ng-template

Démarrer un nouveau projet

ng new nom-de-mon-app (kebab-case) --skip-tests=true --style=scss --routing

COMPONENT (COMPOSANT)

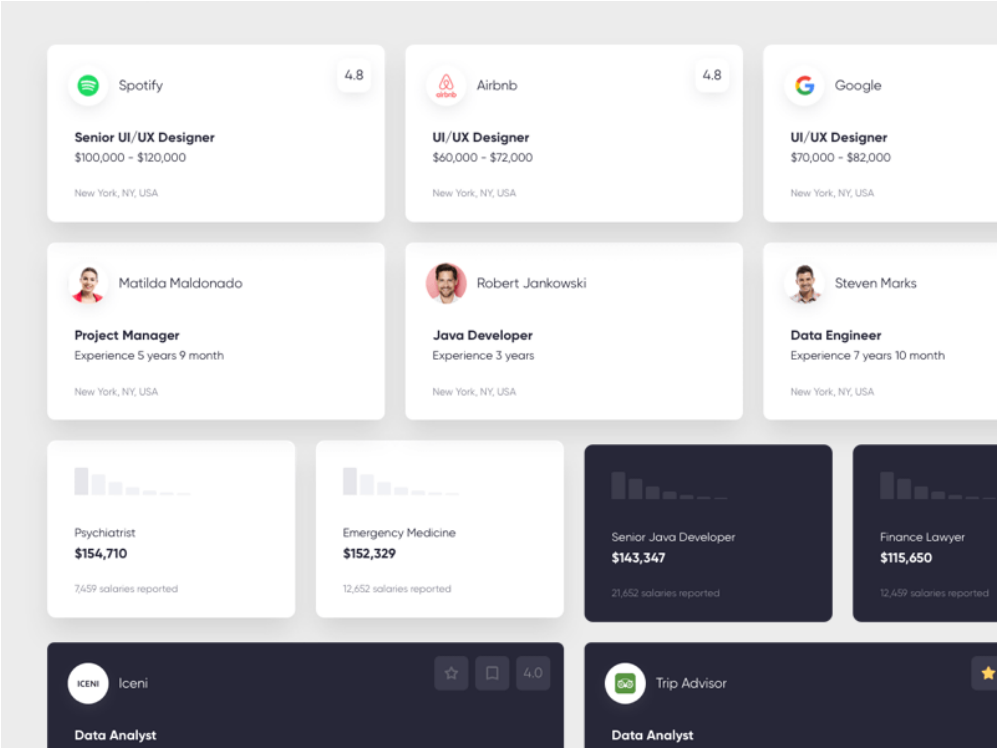


- Web Components

  Les components (ou composants) vont nous permettre de structurer notre application Angular.

  On part toujours d'un composant racine (AppComponent), qui représente l'ensemble de l'application. Il contient des sous-composants, qui représentent par exemple des zones de l'écran comme l'en-tête, le contenu et le pied de page. À leur tour, ces sous-composants peuvent contenir des sous-sous-composants, et ainsi de suite. Cette imbrication de composants s'appelle l'**arbre des composants**.

  Générer un composant ng generate component nom-du-component (kebab-case)
- Un "component" Angular contient:
  - Un **template** contenant l'interface utilisateur en HTML. Nous utiliserons le databinding afin de rendre la vue dynamique ;
  - Une **Classe** (class) contenant le code associé à la vue, des propriétés et méthodes logiques qui seront utilisées dans la vue ;
  - Des **metadata** nous permettant de définir la classe comme étant un composant Angular (component).
- Text interpolation
- Property binding, lire des données depuis le composant
- Event binding, évènements personnalisés
- **Exercice générer un component card**



DIRECTIVES

- Directive : fonctionnement et création

  Les directives sont des classes permettant d'**enrichir et modifier la vue** par simple **ajout d'attributs HTML sur le template**.
- Les directives fournies par Angular
- Attribute directives
- Structural directives
- Directives complexes
- **TP :** Première directive

PIPES

- Les transformateurs fournis
- Formater une chaîne
- Formater des collections
- Utiliser un pipe comme un service
- **TP :** Créer ses propres pipes

ROUTER

- RouterModule: Configuration des routes et URLs

- Définitions des routes, liens et redirection, paramètres
- Hiérarchies de routes
- Vues imbriquées
- Cycle de vie (Routing lifecycle)
- **TP :**

SERVICES

- Les services fournis
- Injection de service
- **TP :** Injecter les services fournis par Angular

OBSERVABLES

- Introduction à RxJS
- Le concept d'Observable
- Les principaux opérateurs
- Lien avec les promesses
- TP : premier pas avec **RXJS**

ÉCHANGER AVEC UN SERVEUR

- Requêtes HTTP
- Communication avec une API
- Afficher des données externes
- Afficher des données asynchrones avec AsyncPipe
- **TP :** récupérer et afficher des données à partir d'une API REST distance

FORMULAIRE

- Créer un formulaire
- Template-driven forms
- ngModel
- Reactive forms
- Validation et erreurs
- Observateurs
- **TP :** Créer ses propres validateurs, envoyer une requête POST

PERFORMANCE

- Astuces, bonnes pratiques & outils
- Découverte des différentes stratégies de compilation
- **TP :** Tester la performance

MODULES

- Déclarations d’un module : imports et exports
- Les providers d’un module