



Ndèye Aïssatou Cissé

Développeuse Web & Mobile Certifiée

Consultante & Formatrice en Développement d'Applications

DÉVELOPPEMENT WEB FRONTEND AVEC ANGULAR

LICENCE 3 INFORMATIQUE DE GESTION

PRÉREQUIS DE LA FORMATION

- ❖ Comprendre le fonctionnement du Web
- ❖ HTML
- ❖ CSS
- ❖ JavaScript

PROGRAMME DE LA FORMATION

- 1^{ère} Partie: **Débutez avec Angular**
- 2^{ème} Partie: **Les fondamentaux du framework Angular**
- 3^{ème} Partie: **Services et routage en Angular**
- 4^{ème} Partie: **La communication avec le serveur**
- 5^{ème} Partie: **Les formulaires en Angular**

DÉBUTEZ AVEC ANGULAR

LICENCE 3 INFORMATIQUE DE GESTION

PLAN

- I.** Rappel sur les fondamentaux du Web
 - II.** Présentation du framework Angular
 - III.** Téléchargement et installation des outils de travail
 - IV.** Création et architecture d'un projet Angular
- Cas pratique: Mise en page statique avec Angular



I-RAPPEL SUR LES FONDAMENTAUX DU WEB

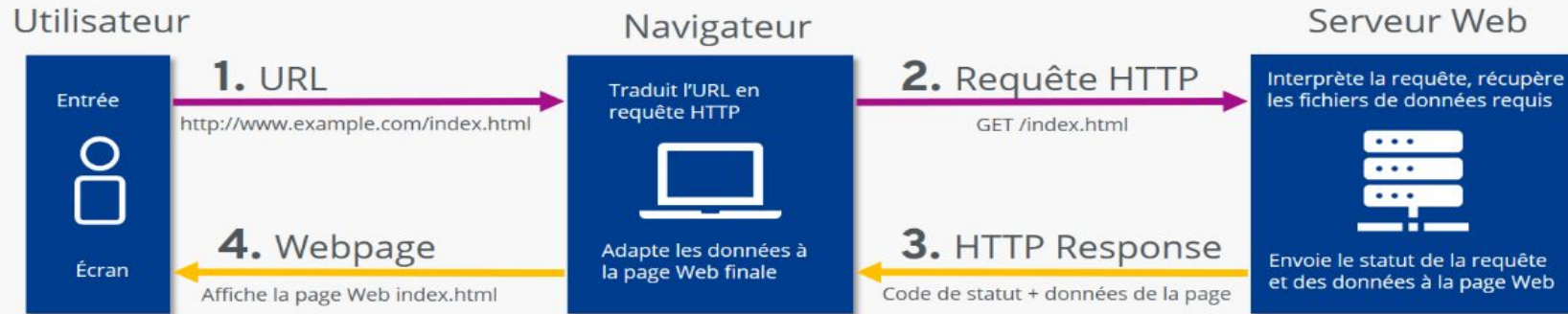
1. QU'EST-CE QUE LE WEB?

- ❖ Le **Web**, également connu sous le nom de **World Wide Web (WWW)**, est un système de documents hypertextes accessibles via Internet.
- ❖ Il permet aux utilisateurs de naviguer entre des pages Web à l'aide de liens hypertextes.
- ❖ Inventé en 1989 par Tim Berners-Lee, le Web a révolutionné la manière dont l'information est partagée et consommée à travers le monde.

2. LA COMMUNICATION CLIENT – SERVEUR: LE PROTOCOLE HTTP


- Le protocole **HTTP (Hypertext Transfer Protocol)** est un protocole de communication utilisé pour le transfert de données sur le **World Wide Web**. Il définit la structure et les règles pour la communication entre un **client** (tel qu'un navigateur web) et un **serveur web**.

Processus de communication HTTP



3. DIFFÉRENCE ENTRE UN SITE WEB ET UNE APPLICATION WEB

- Le rôle principal d'un **site Web** est de fournir et présenter de l'information aux visiteurs, par exemple un blogue, un site de nouvelles ou un site d'information; on dit que le **site est statique**.
- Tandis qu'une **application Web** est tout site web qui permet à ses utilisateurs d'accomplir des tâches spécifiques; on dit que le **site est dynamique**.
 - Une **application Web** a 3 responsabilités:
 - Récupérer l'information
 - Traiter ces informations
 - Retourner une réponse

A decorative wavy line in light blue and white on the left side of the slide.

II- PRÉSENTATION DU FRAMEWORK ANGULAR

1. QU'EST-CE QUE C'EST ANGULAR?

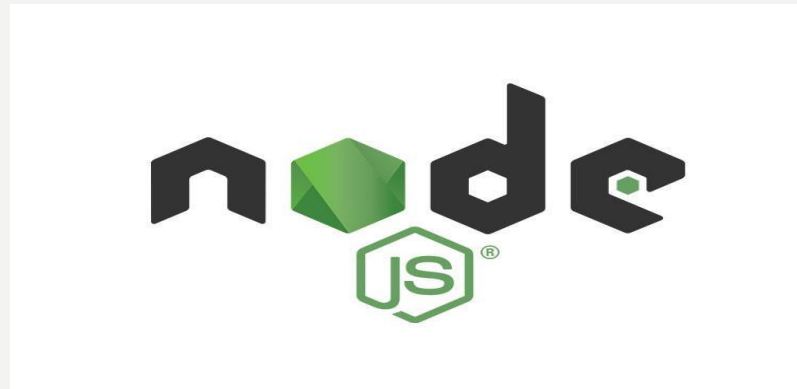
- ❖ **Angular est un cadre de travail (framework en anglais) JavaScript développé par Google, conçu pour la création d'applications Web robustes et puissantes. Il utilise HTML, CSS et TypeScript qui sera compilé en JavaScript afin d'être compris par les navigateurs. Angular offre une gamme complète de fonctionnalités pour le développement Web moderne.**
- ❖ **La première version du framework, connue sous le nom d'AngularJS, a été publiée en 2009. Par la suite, Angular 2 (ou simplement Angular) a été lancé en mars 2017, avec une refonte majeure du framework par rapport à AngularJS. Angular a introduit de nouvelles fonctionnalités et une architecture plus performante, tout en conservant certains concepts clés de son prédécesseur.**
- ❖ **Depuis lors, Angular est régulièrement mis à jour, avec une nouvelle version majeure publiée environ tous les six mois. Chaque nouvelle version apporte des améliorations, des fonctionnalités et des corrections de bogues, ce qui permet aux développeurs de rester à jour avec les dernières technologies et bonnes pratiques de développement Web.**
- ❖ **Dans cette formation, nous allons travailler avec la version 17.**

2. AUTRES ALTERNATIVES À ANGULAR



A decorative wavy line in light blue and white on the left side of the slide.

III- TÉLÉCHARGEMENT ET INSTALLATION DES OUTILS DE TRAVAIL



NODE.JS ET NPM

❖ Qu'est-ce que Node.js?

- **Node.js est un environnement d'exécution JavaScript côté serveur.** Il permet d'exécuter du code JavaScript en dehors d'un navigateur web.
- Contrairement à JavaScript traditionnellement exécuté dans les navigateurs pour créer des applications web interactives, Node.js permet d'utiliser JavaScript pour développer des applications côté serveur, comme des serveurs web, des API, et des outils en ligne de commande.
- **Node.js est livré avec npm.**


❖ Qu'est-ce que NPM?

- **NPM (Node Package Manager) est le gestionnaire de paquets officiel pour l'environnement d'exécution Node.js.** C'est un outil essentiel pour les développeurs JavaScript, car il permet d'installer, de gérer les dépendances des projets et de partager des modules de code avec la communauté de développeurs JavaScript.

ANGULAR CLI

❖ Qu'est-ce que c'est Angular CLI?

- Angular CLI est un outil de ligne de commande qui vous permet d'échafauder, de développer, de tester, de déployer et de maintenir des applications Angular directement à partir d'un shell de commande.
- Angular CLI simplifie le processus de développement en automatisant de nombreuses tâches courantes et en fournissant des commandes pour gérer le cycle de vie complet des applications Angular, de la création à la production.
- Angular CLI est publié sur npm en tant que package `@angular/cli`. Lors de l'installation de ce package, il inclut un binaire nommé `ng`.
- `ng` est l'exécutable (le binaire) qui permet d'interagir avec Angular CLI. Vous utilisez cette commande pour créer, développer, tester et déployer des applications Angular.

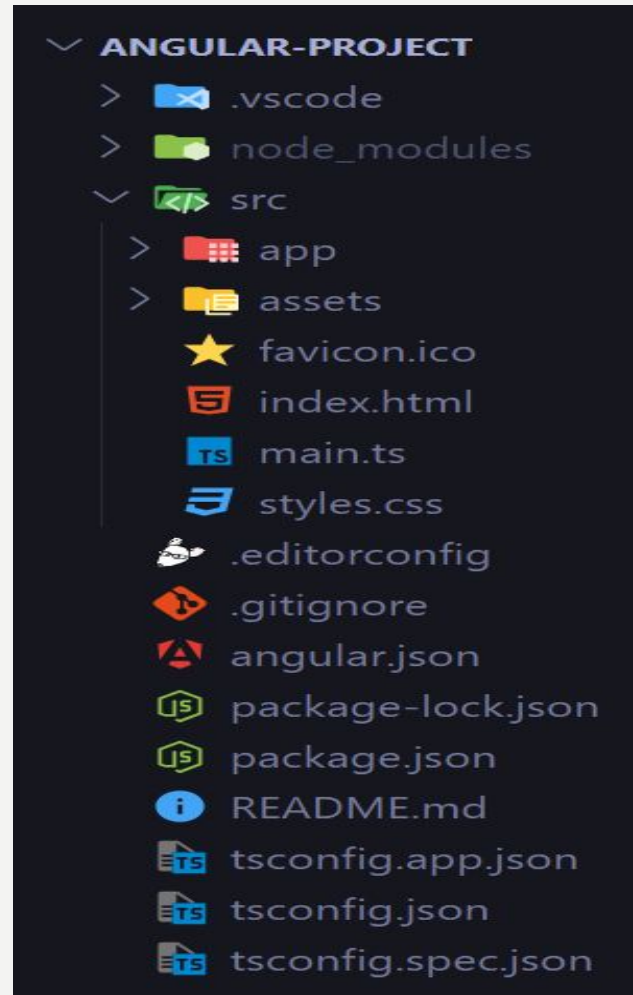
A decorative wavy line in light blue and white, flowing from the top left towards the bottom left of the slide.

IV- CRÉATION ET ARCHITECTURE D'UN PROJET ANGULAR

1. CRÉATION D'UN PROJET AVEC ANGULAR CLI

- ❖ Création de projet avec Angular CLI:
 - **ng new project_name**
- ❖ Démarrer le serveur de développement:
 - **cd project_name**
 - **ng serve**

2. COMPRENDRE LE CONTENU D'UN PROJET ANGULAR (1/4)



2. COMPRENDRE LE CONTENU D'UN PROJET ANGULAR (2/4)

- **.editorconfig:** Ce fichier contient des configurations pour l'éditeur de code, telles que le style de codage, l'indentation, etc. Il permet de maintenir la cohérence du code entre différents développeurs travaillant sur le projet.
- **.gitignore:** Ce fichier spécifie les fichiers et dossiers qui doivent être ignorés par Git lors de la gestion du code source. Cela inclut généralement les fichiers générés, les dépendances et les fichiers sensibles qui ne doivent pas être suivis dans le référentiel Git.
- **angular.json:** Fichier de configuration spécifique à Angular CLI. Il contient des configurations telles que les chemins des fichiers sources, les paramètres de compilation, les options de construction, etc.
- **package-lock.json:** Ce fichier est généré automatiquement par npm et contient des informations détaillées sur les versions spécifiques des dépendances installées dans le projet. Il garantit que les versions des dépendances restent cohérentes entre les différentes installations.
- **package.json:** Ce fichier contient des métadonnées sur le projet, y compris les dépendances utilisées, les scripts de démarrage, les versions, les licences, etc. Il est utilisé par npm pour installer les dépendances du projet et exécuter des scripts.
- **README.md:** Ce fichier contient la documentation du projet, y compris des informations sur son utilisation, son installation, ses fonctionnalités, sa contribution, etc. Il est souvent affiché sur la page d'accueil du référentiel GitHub du projet.

2. COMPRENDRE LE CONTENU D'UN PROJET ANGULAR (3/4)

- **tsconfig.json:** Fichier de configuration qui permet de faire la compilation de TypeScript en JavaScript. Il spécifie les options de compilation TypeScript telles que la version de JavaScript cible, les paramètres de module, les chemins des fichiers sources, etc.
- **tsconfig.app.json:** Ce fichier étend la configuration de tsconfig.json spécifiquement pour l'application Angular. Il peut contenir des options de compilation spécifiques à l'application.
- **tsconfig.spec.json:** Fichier de configuration spécifique pour les tests. Il permet de configurer les options de compilation TypeScript pour les tests unitaires et d'intégration.
- **node_modules:** Ce dossier contient toutes les dépendances du projet, y compris les bibliothèques tierces et les modules nécessaires au fonctionnement de l'application.
- **src:** Ce dossier contient tout le contenu source de l'application Angular, y compris les composants, les services, les modèles, les styles, etc.

2. COMPRENDRE LE CONTENU D'UN PROJET ANGULAR (4/4)

- **src/app:** Ce dossier contient le composant principal de l'application Angular (AppComponent), ainsi que tous les autres composants, services et directives utilisés dans l'application.
- **src/assets:** Ce dossier contient des fichiers statiques tels que des images, des fichiers CSS ou des fichiers JSON utilisés par l'application.
- **src/favicon.ico:** Ce fichier contient l'icône de favicon de l'application, qui s'affiche dans l'onglet du navigateur.
- **src/index.html:** Ce fichier représente la première page HTML chargée par le navigateur lorsqu'un utilisateur accède à l'application Angular. Il s'agit du point d'entrée initial où le navigateur commence à rendre le contenu de l'application. Il contient généralement des balises HTML de base ainsi que des balises spéciales (**<app-root>**) qui servent de point d'ancrage pour l'application Angular.
- **src/main.ts:** Ce fichier est le point d'entrée du code TypeScript de l'application Angular c'est-à-dire que c'est là où commence l'exécution de l'application. Il charge le composant principal (AppComponent) et le fichier de configuration principal appConfig qui contient les fonctionnalités Angular nécessaires au démarrage de l'application.
- **src/styles.css:** Ce fichier contient les styles globaux de l'application, qui s'appliquent à toutes les composantes de l'application.

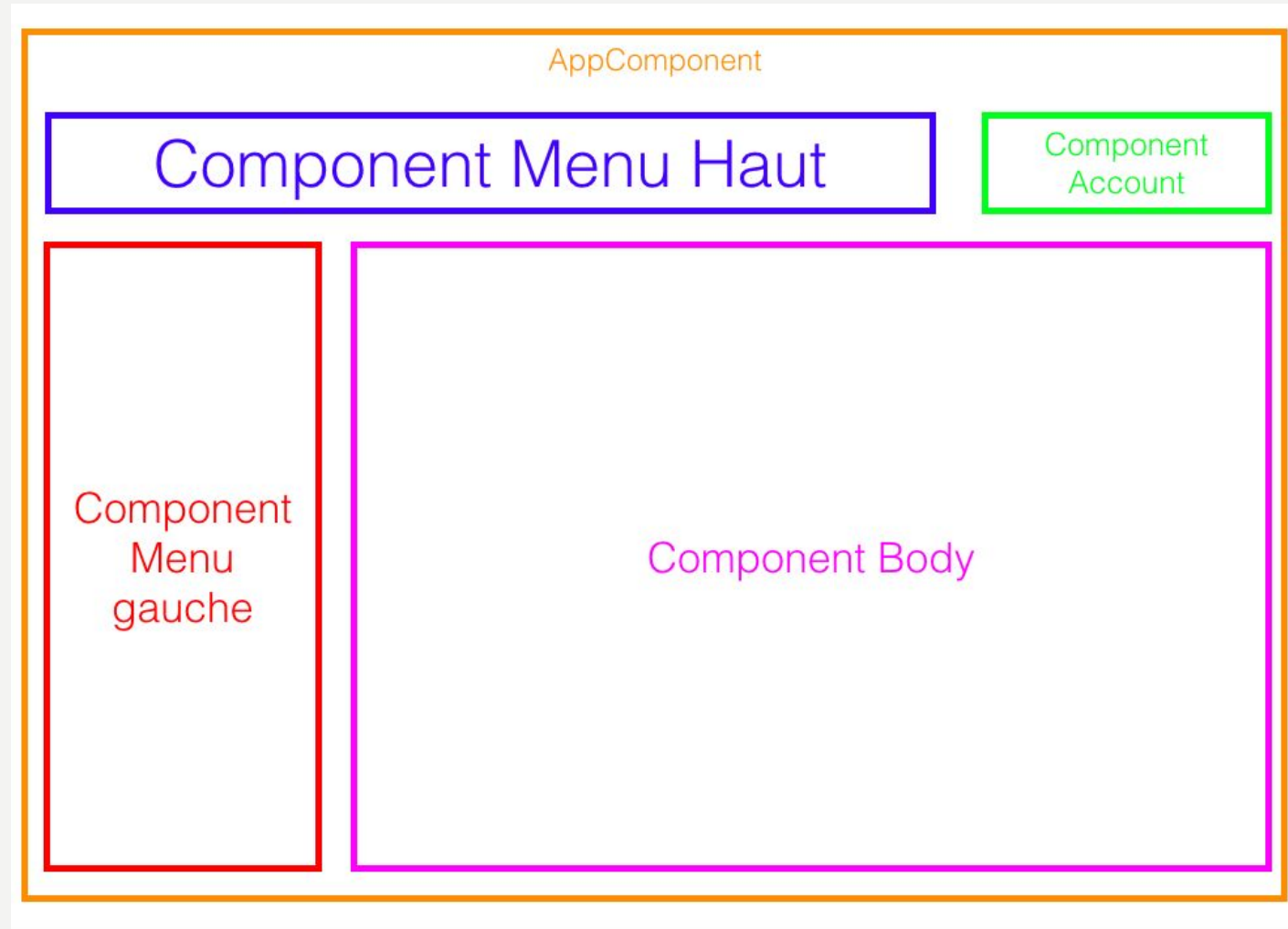
3. ARCHITECTURE D'UN PROJET ANGULAR

- Qu'est-ce que c'est une application Angular?
 - Une **application Angular** est une application Web de type **Single Page Application (SPA)**, ce qui signifie qu'elle charge **une seule page HTML** initialement et interagit avec le navigateur en modifiant dynamiquement le contenu de cette page au fur et à mesure de l'utilisation de l'application, sans nécessiter de rechargement complet de la page.
 - Lorsqu'un utilisateur accède à l'application Angular, le navigateur charge initialement le fichier **index.html**, qui sert de point d'entrée de l'application Angular.
 - Angular est ensuite chargé et initialisé à partir du fichier **main.ts**, qui agit en tant que point d'entrée du code TypeScript de l'application. Ce fichier est responsable de démarrer l'application Angular en chargeant le composant principal (AppComponent)
 - Et c'est **AppComponent** qui va charger tous les autres composants qui seront créés durant le projet.

4. NOTION DE WEB COMPONENT (1/2)

- **Angular repose sur les Web Components et utilise l'architecture MVVM (Model View ViewModel), qui s'avère être particulièrement adaptée pour les applications réactives.**
- **Un composant (ou component en anglais)** représente une unité essentielle dans Angular, combinant un **modèle** (les données structurées transmises à la vue), une **vue** (l'affichage de ces données) et un **contrôleur** (qui gère la logique de traitement des données affichées dans la vue).
- **L'AppComponent est le composant principal d'une application Angular, définissant la structure globale de la page Web. Tous les autres composants de l'application sont imbriqués à l'intérieur de celui-ci.**

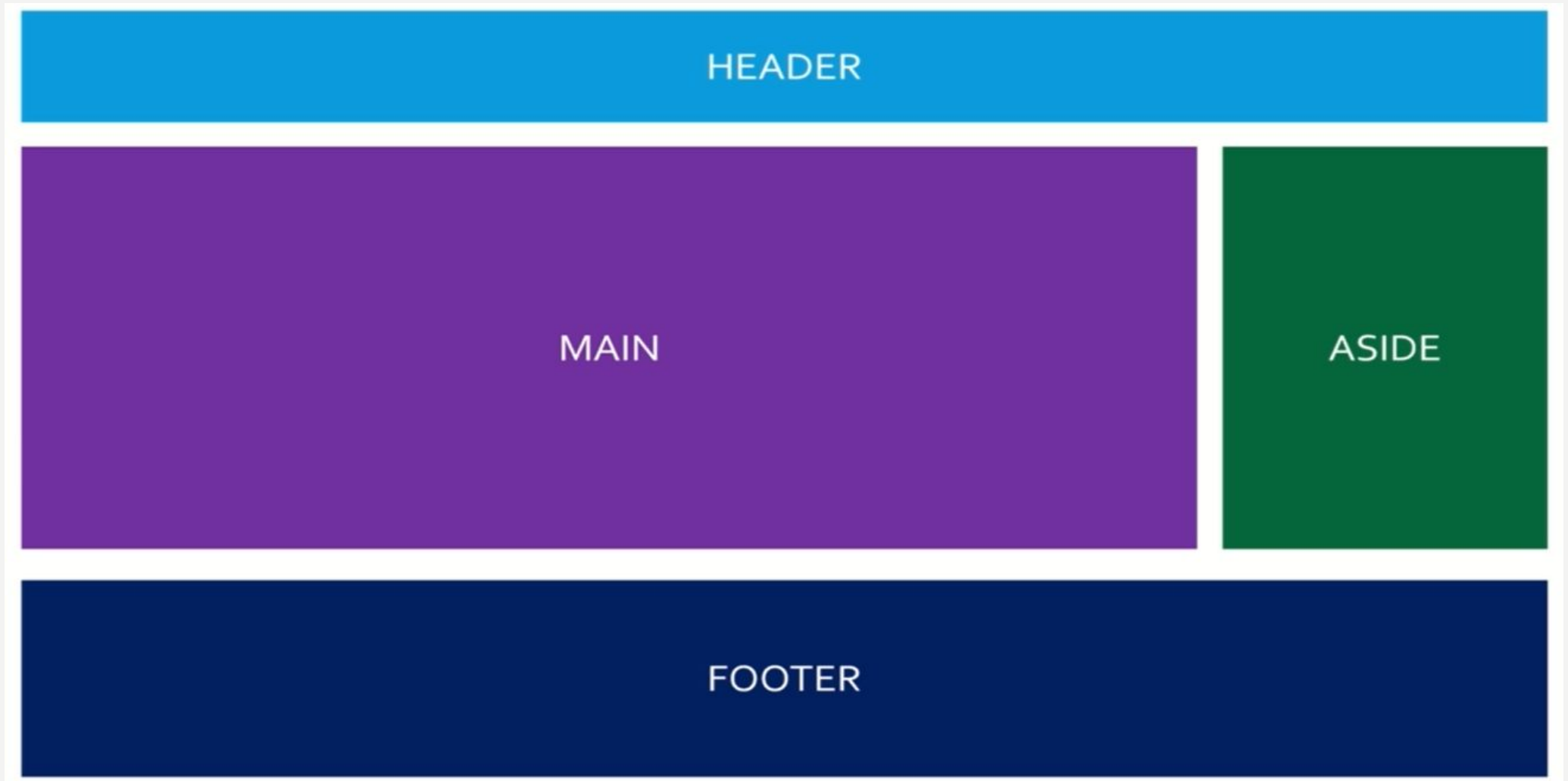
4. NOTION DE WEB COMPONENT (2/2)



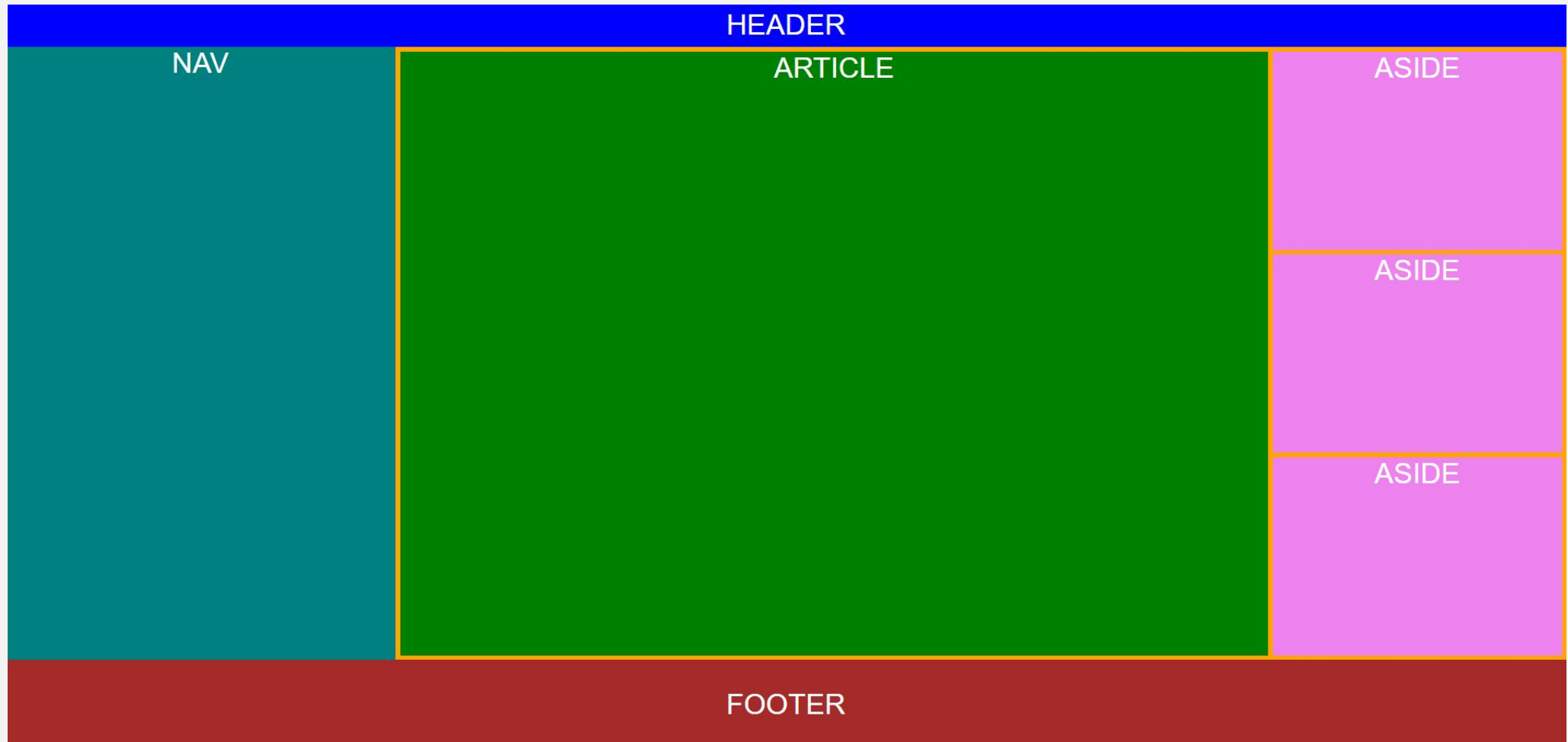
A decorative wavy line in light blue and white on the left side of the slide.

CAS PRATIQUE: MISE EN PAGE STATIQUE AVEC ANGULAR

TP1: MISE EN PAGE STATIQUE SIMPLE



TP2: MISE EN PAGE STATIQUE AVANCÉ



LES FONDAMENTAUX DU FRAMEWORK ANGULAR

LICENCE 3 INFORMATIQUE DE GESTION

PLAN

- I. Découverte de TypeScript
- II. Databinding (liaison de données)
- III. Les directives en Angular
- IV. La communication entre composants

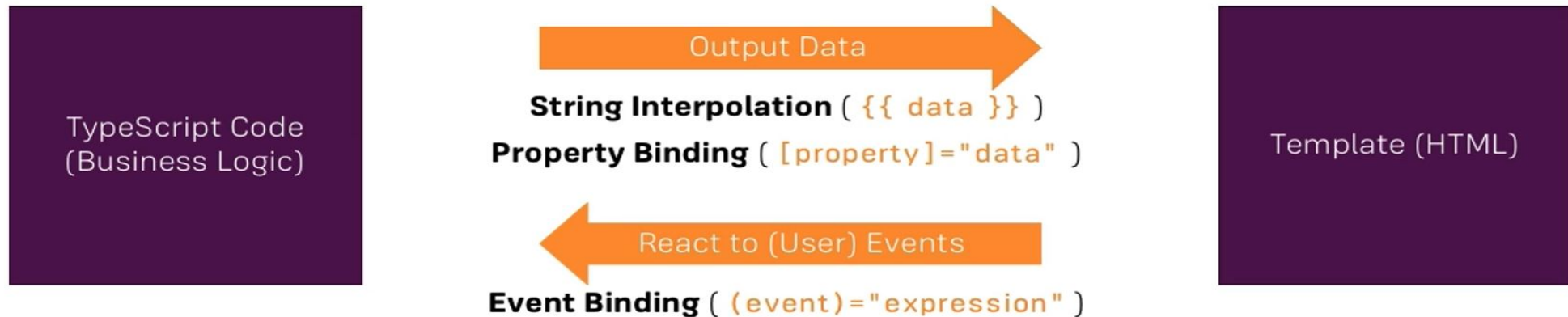
I- DÉCOUVERTE DE TYPESCRIPT

- ❖ TypeScript est un sur-ensemble de JavaScript qui ajoute des fonctionnalités de typage statique et de développement orienté objet, ce qui rend le langage plus robuste et maintenable. Développé par Microsoft, TypeScript se compile en JavaScript, ce qui signifie qu'il peut être exécuté dans n'importe quel environnement qui supporte JavaScript, comme les navigateurs et Node.js.
- ❖ Angular utilise TypeScript pour ses avantages en termes de typage statique, de fonctionnalités orientées objet, de meilleure lisibilité et maintenabilité du code, de compatibilité avec les futures versions de JavaScript, et d'un excellent écosystème d'outils. TypeScript aide à créer des applications robustes et maintenables, ce qui est particulièrement bénéfique pour les projets à grande échelle comme ceux développés avec Angular.

II- DATABINDING (LIAISON DE DONNÉES)

Understanding Databinding

Databinding = Communication



Combination of Both: **Two-Way-Binding** (`[(ngModel)]="data"`)

III- LES DIRECTIVES EN ANGULAR (1/2)

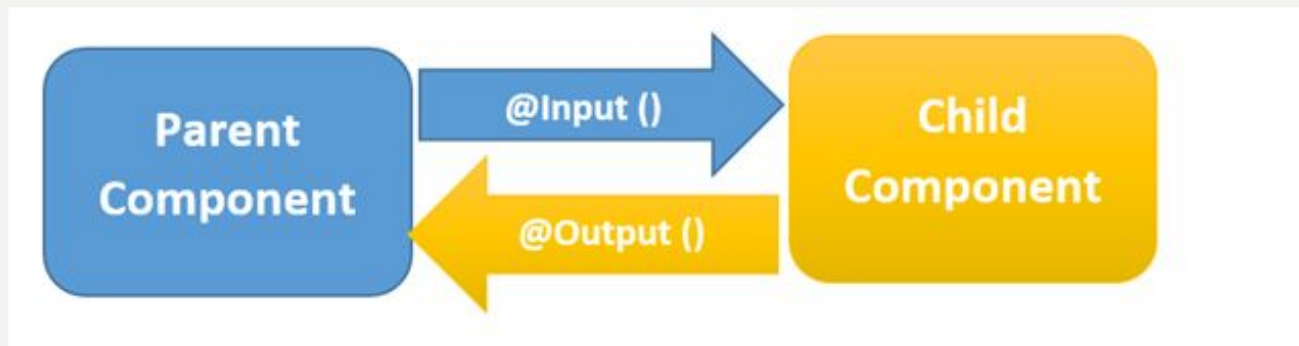
- ❖ Dans une application Angular, une **directive** est un élément qui va nous permettre depuis le template HTML, d'interagir directement avec le DOM (Document Object Model) de notre page web.
- ❖ Angular propose deux types de directives intégrées: **les directives d'attribut** et les **directives structurelles**.
- ❖ **Les directives d'attribut** sont utilisées pour modifier le comportement, les attributs ou les propriétés des éléments HTML. Elles sont appliquées comme des attributs HTML normaux et permettent de manipuler le DOM de manière efficace. Elles offrent un moyen puissant de contrôler le comportement et l'apparence des éléments HTML dans une application Angular.
- ❖ Les directives d'attribut les plus courantes sont les suivantes:
 - **NgClass:** permet d'ajouter ou de supprimer dynamiquement des classes CSS à un élément en fonction d'une condition ou d'une expression.
 - **NgStyle:** permet d'appliquer dynamiquement des styles CSS à un élément en fonction de valeurs d'expression.
 - **NgModel:** elle est utilisée pour créer une liaison bidirectionnelle entre les données dans le modèle de votre composant et les contrôles de formulaire HTML, permettant ainsi aux utilisateurs de saisir des données dans les formulaires et de les refléter dans le modèle et vice versa.

III- LES DIRECTIVES EN ANGULAR (2/2)

- ❖ **Les directives structurelles** en Angular sont responsables de la manipulation de la structure du DOM. Elles permettent de conditionner l'affichage ou la modification de certaines parties du DOM en fonction de conditions ou de données dynamiques.
- ❖ Les directives structurelles les plus courantes sont les suivantes:
 - **NgIf:** Cette directive est utilisée pour conditionner l'affichage d'un élément en fonction d'une expression booléenne. Si l'expression est évaluée à true, l'élément est affiché ; sinon, il est retiré du DOM.
 - **NgFor:** Cette directive est utilisée pour répéter un élément du DOM pour chaque élément d'une liste. Elle crée ainsi une répétition de la structure HTML pour chaque élément de la liste.
 - **NgSwitch:** Cette directive permet de basculer entre différentes vues alternatives en fonction de la valeur d'une expression. Il est souvent utilisé pour remplacer plusieurs instructions NgIf imbriquées.
- ❖ On note également la possibilité de **créer nos propres directives (directives personnalisées)** destinées à effectuer une tâche particulière.

IV- LA COMMUNICATION ENTRE COMPOSANTS

- ❖ Une application Angular typique est composée de nombreux composants, organisés de manière hiérarchique. La communication entre ces composants est cruciale pour garantir une application cohérente et fonctionnelle. L'enjeu principal de cette **communication** réside dans la **transmission efficace des données et des événements entre composants**. Cette transmission peut se produire dans plusieurs directions parmi lesquelles:
 - **Communication Parent-Enfant (@Input)**: Un composant parent doit transmettre des données à un composant enfant.
 - **Communication Enfant-Parent (@Output)**: Un composant enfant doit informer son composant parent d'un événement ou d'un changement de données.



SERVICES ET ROUTAGE EN ANGULAR

LICENCE 3 INFORMATIQUE DE GESTION

PLAN

- I.** Notion de Services
- II.** Services et Injection de Dépendances
- III.** Routage et Navigation
- IV.** Routes paramétrées

1. NOTION DE SERVICES

- ❖ Un service dans Angular est une classe TypeScript conçue pour encapsuler des fonctionnalités réutilisables, qui peuvent être partagées entre différents composants.
- ❖ Les services permettent de séparer la logique métier ou les fonctionnalités partagées, comme l'accès aux données, la gestion des états de l'application, ou les fonctionnalités utilitaires, du code des composants.
- ❖ **Caractéristiques des Services:**
 - **Réutilisabilité:** Les services peuvent être utilisés par plusieurs composants, ce qui réduit la duplication de code.
 - **Séparation des Préoccupations:** En déplaçant la logique métier dans les services, les composants deviennent plus simples et plus faciles à gérer.
 - **Testabilité:** Les services peuvent être testés indépendamment des composants, facilitant ainsi les tests unitaires.

2. SERVICES ET INJECTION DE DÉPENDANCES

- ❖ L'injection de dépendances (DI, Dependency Injection) est un concept central dans Angular qui permet de fournir des instances de services à différents composants ou autres services de manière automatisée et cohérente. Cette approche améliore la modularité, facilite la gestion des dépendances et simplifie les tests unitaires.
- ❖ Le principe de l'injection de dépendances repose sur la séparation des responsabilités. Plutôt que de laisser les composants créer leurs propres dépendances, ils les reçoivent de l'extérieur, généralement via leurs constructeurs.

3. ROUTAGE ET NAVIGATION

- ❖ Le **routing** en Angular permet de créer des **applications à page unique SPA (Single Page Application)** en naviguant entre différentes vues ou composants sans recharger la page. **@angular/router** est le **module responsable de la gestion des routes**. Il permet de définir des **chemins de navigation**, de **gérer les paramètres d'URL**, de **protéger les routes** et de **configurer des redirections**.
- ❖ **Principes de Base du Routing:**
 - ✓ **Définition des Routes:** Les routes sont définies dans un tableau d'objets de routes, chaque objet représentant une route spécifique avec un chemin et un composant associé.
 - ✓ **RouterModule:** Pour utiliser le routing, le RouterModule d'Angular doit être importé et configuré avec les routes définies.
 - ✓ **RouterOutlet:** Une directive utilisée dans les templates pour indiquer où le contenu correspondant à la route active doit être affiché.
 - ✓ **RouterLink:** Une directive utilisée pour créer des liens de navigation entre les différentes routes.

4. ROUTES PARAMÉTRÉES

- ❖ Les routes paramétrées sont des fonctionnalités essentielles dans le module de routage d'Angular. Elles permettent de créer des applications dynamiques en gérant les paramètres de l'URL.
- ❖ Les routes paramétrées permettent de passer des informations dans l'URL, comme des identifiants ou des filtres de recherche. Cela est utile pour afficher des détails spécifiques basés sur un identifiant ou d'autres paramètres.

LA COMMUNICATION AVEC LE SERVEUR

LICENCE 3 INFORMATIQUE DE GESTION

PLAN

- I. Découverte de HttpClient
- II. Les opérations CRUD
- III. La Programmation Réactive

1. DÉCOUVERTE DE HTTPCLIENT

- ❖ **HttpClient** est un outil puissant du framework Angular qui permet de communiquer avec des serveurs backend via des requêtes HTTP. Ce module fournit une API facile à utiliser pour effectuer des opérations CRUD (Create, Read, Update, Delete) et pour travailler de manière réactive avec les flux de données.
- ❖ Pour utiliser HttpClientModule, vous devez d'abord l'importer dans le fichier de configuration **app.config.ts** comme provider:

```
providers: [  
  provideHttpClient()  
]
```

2. LES OPÉRATIONS CRUD

- ❖ CRUD représente les opérations fondamentales pour manipuler les données : Create (Créer), Read (Lire), Update (Mettre à jour) et Delete (Supprimer).
- ❖ HttpClient fournit des méthodes pour chaque opération présentées ci-après:
 - **Create (POST):** Création d'une nouvelle ressource avec la méthode POST.
 - **Read (GET):** Lire ou récupérer des données avec la méthode GET.
 - **Update (PUT/PATCH):** Mettre à jour une ressource avec les méthodes PUT ou PATCH.
 - **Delete (DELETE):** Supprimer une ressource avec la méthode DELETE.

3. LA PROGRAMMATION RÉACTIVE (1/2)

- ❖ **La programmation réactive** est un paradigme de programmation qui se concentre sur **les flux de données et la propagation des changements**. Elle permet de réagir automatiquement aux modifications des données en temps réel, ce qui est particulièrement utile pour les interfaces utilisateur dynamiques où les changements de données doivent entraîner des mises à jour immédiates de l'affichage.
- ❖ Ce paradigme est particulièrement utile dans les applications modernes où les interactions utilisateur et les communications réseau sont fréquentes, rendant la gestion des états asynchrones complexe. La programmation réactive facilite cette gestion en permettant de manipuler et de composer des flux de données asynchrones à l'aide d'observables.
- ❖ En Angular, la bibliothèque **RxJS (Reactive Extensions for JavaScript)** fournit les outils nécessaires pour appliquer les concepts de la programmation réactive. RxJS permet de créer et de manipuler des observables, d'appliquer des opérateurs pour transformer les flux de données et de gérer les abonnements pour réagir aux nouvelles données de manière fluide et efficace.

3. LA PROGRAMMATION RÉACTIVE: CONCEPTS CLÉS (2/2)

- ❖ En Angular, la programmation réactive est largement utilisée pour gérer les données asynchrones et les événements, notamment grâce à la bibliothèque RxJS.
- ❖ Voici quelques concepts clés de la programmation réactive :
 - 1. Flux de données (Streams):** Ce sont des séquences d'événements ordonnés dans le temps. Ils peuvent émettre des données de manière continue, et les composants réactifs peuvent écouter ces flux pour réagir aux nouvelles données.
 - 2. Observables:** Ce sont des objets qui émettent des flux de données. Ils sont observés par des observateurs (ou abonnés) qui réagissent à ces données.
 - 3. Observateurs (Subscribers):** Ce sont des objets ou fonctions qui s'abonnent aux observables et réagissent aux nouvelles données émises par ces derniers.
 - 4. Opérateurs:** Ce sont des fonctions qui permettent de manipuler et de transformer les flux de données. Par exemple, les opérateurs peuvent filtrer, mapper, concaténer ou combiner des flux de données.
 - 5. Propagation des changements:** Lorsque les données changent, ces changements se propagent automatiquement à travers les flux de données, déclenchant des mises à jour dans les composants qui dépendent de ces données.

LES FORMULAIRES EN ANGULAR

LICENCE 3 INFORMATIQUE DE GESTION

PLAN

- I. Template Forms
- II. Reactive Forms

Angular offre deux approches principales pour la gestion des formulaires :

les **Template Forms** et les **Reactive Forms**.

Les Template Forms sont simples et rapides à configurer pour des formulaires basiques, tandis que les Reactive Forms offrent une puissance et une flexibilité accrues pour les applications complexes nécessitant des validations et une gestion d'état avancées. Le choix entre les deux dépend de la complexité des formulaires et des besoins spécifiques de l'application.

1. TEMPLATE FORMS

- ❖ Les **Template Forms** (ou **Template-Driven Forms**) reposent sur des directives Angular intégrées dans le template HTML pour gérer les formulaires. Ils sont plus simples à configurer et conviennent aux formulaires simples. Pour utiliser les Templates Forms, il faut importer le module **FormsModule**.
- ❖ **Caractéristiques des Template Forms:**
 - ❑ **Basés sur les Templates HTML:** Les Template Forms utilisent les directives Angular directement dans le HTML, ce qui permet de gérer les états des champs de formulaire (comme la validation et la soumission) via le DOM.
 - ❑ **To-Way Data Binding:** Avec les Template Forms, les données des formulaires sont automatiquement synchronisées avec le modèle de données de l'application à l'aide de la directive **ngModel**.
 - ❑ **Validation Déclarative:** Les validations sont définies directement dans le template HTML en utilisant des attributs spécifiques comme **required**, **minlength**, **maxlength**, etc.

2. RÉACTIVE FORMS

- ❖ Les **Reactive Forms** (ou **Modèle-Driven Forms**) offrent une approche plus structurée et puissante pour gérer les formulaires. Ils sont basés sur une programmation réactive utilisant le module **ReactiveFormsModule**.
- ❖ **Caractéristiques des Reactive Forms:**
 - **Basés sur des Modèles:** Les Reactive Forms sont construits et gérés par le code TypeScript, offrant un meilleur contrôle sur la structure et les validations des formulaires.
 - **Validation Programmatique:** La validation des formulaires est définie de manière programmatique, ce qui facilite les tests unitaires et la réutilisation des règles de validation.
 - **Programmation Réactive:** Utilisation de la bibliothèque RxJS pour gérer les flux de données et les événements, ce qui permet une gestion fine des états des formulaires.

Let's
practice!