I - Présentation de Angular :

Doc : <https://angular.dev/>

JS est un langage orienté prototype, et non objet oui il manipule des objets mais il est orienté prototype. Cependant typescrypt est orienté objet.

IONIC : permet de créer des applications mobiles hybrides avec angular.

JS : est un langage interprété

TS : est un langage compilé

Le navigateur ne lit que le JS

Strict mode en JS : permet de rajouter des contraintes au développeur, il permet d’ empêcher le développeur de faire des erreurs (en gros il lève des exception visuelles pour le développeur en cas d’erreur)

* Une application angular c’est un arbre de modules
  + Le module racine est **app.module.ts**, il est bootstrappé (lancer en premier) dans le fichier **main.ts**
* Un module c’est un ou plusieurs arbres de composants
  + Le composant racine est **app.component.ts**, il est bootstrappé dans le module **app.module** dans le tableau **bootsrap** qui se trouve dans l’annotation **@NgModule**
* Un composant est composé d’une vue (.html) et d’un modèle la communication entre le modèle et la vue se faire par le binding

II - Les composants :

* Un composant parent connaît ses composants enfants (on peut accéder aux composants enfants avec un ctrl + clic sur le composant enfant depuis la vue parent) c’est pour cela qu’on utilise le property binding pour passer des propriétés au composant enfant depuis le composant parent.
* Un composant enfant ne connaît pas ces composants parents, du on utilise un event binding, le composant enfant émet un évènement et les parents se connectent à cet évènement

Quelle est la différence entre ngOnInit et le constructeur ?

Le constructeur est un principe de la POO, il permet de l’injection d’indépendance.

ngOnInit est une convention spécifique à angular qui permet de gérer le cycle de vie des composants

III - Les directives :

* Les composants : les composants sont aussi des directives
* Les directives structurelles : elles modifient la structure du DOM (if, for, switch, etc.)
* Les directives d’attributs : elles modifient les attributs (ngStyle, ngClass, etc.)

IV - Les Services :

Quand vous avez un ensemble de composants qui injectent le même service, ce service doit être provider dans l’un de leur parent en commun. Par défaut lors de la création d’un service avec le CLI on le provide dans le root de l’appli c’est-à-dire dans l’app-module :

@Injectable({  
 providedIn: 'root',  
})

Ceci permet d’utiliser la même instance du service à chaque fois qu’elle est injectée dans un composant ou un autre service dès lors qu’ils ont tous en commun la racine app.module

@Injectable() permet d’injecter une classe dans une autre classe d’où sa présence dans les services.

Les classes avec des annotations spécifiques telles que les composants (@Component) qui n’ont pas l’annotation @Injectable() implique que les annotations de ces classes possèdent déjà Injectable dans le code source de angular.

V - Les routes

En JS les routes dynamiques doivent être toujours après les routes statiques.

<router-outlet></ router-outlet > est utilisé dans tous les composants qui gèrent la route de ses enfants

VI - Les Observables :

Permet d’exécuter une tâche asynchrone, comme les Promise, cependant un observable est un flux de valeur, contrairement aux promesses qui exécute une tâche bien spécifique en une fois.

Observer permet de gérer ce flux de valeurs en s’abonnant au flux avec subscribe qui implémente les méthodes suivantes qui sont des méthode de l’Observer:

Next() pour insérer une nouvelle valeur dans le flux (obligatoire)

Error() pour insérer une erreur, ferme le flux (facultatif)

Complete() pour terminer le flux (facultatif)

VII – Les intercerptors

Les méthodes HttpRequest, HttpResponse, sont immuables, il ne peuvent pas être modifiées après leur initialisations.

VIII – Les guards

* CanActivate
* CanActivateChild
* CanDeActivate