🚀 RÉSUMÉ RAPIDE QCM ANGULAR – Concepts, Réponses & Pièges

> **Comment lire ce fichier ?**
> Chaque ligne = 1 question. Format : **CONCEPT** → Réponse correcte → ⚠ Piège

📘 SECTION 1 – Cours Angular (40 Questions)

#	Concept → Réponse → Piège
1	**Décorateur composant** → `@Component()` → ⚠ @Component ≠ @Directive (composant = directive AVEC template)
2	**Fichier des routes** → `app-routing.module.ts` → ⚠ index.html = point d'entrée statique, pas routing
3	**Affichage conditionnel** → `*ngIf` → ⚠ L'astérisque (*) est OBLIGATOIRE (directive structurelle)
4	**Définir un service** → `@Injectable()` → ⚠ @Service() N'EXISTE PAS en Angular (c'est Spring/Java)
5	**Transformer données RxJS** → `map()` → ⚠ subscribe() n'est PAS un opérateur, c'est le déclencheur
6	**Hook après constructeur** → `ngOnInit` → ⚠ Constructeur = TS, ngOnInit = Angular. Pas de HTTP dans constructor!
7	**Parent vers enfant** → `@Input()` → ⚠ Confondre Input (vers l'intérieur) et Output (vers l'extérieur)
8	**Two-Way Binding** → `[(ngModel)]` → ⚠ Exige FormsModule + syntaxe "banana in a box" [( )]
9	**Polyfills.ts** → Compatibilité navigateurs anciens → ⚠ Moins critique aujourd'hui (Evergreen browsers)
10	**ANGULAR en minuscules** → pipe `lowercase` → ⚠ Les pipes ne modifient PAS la variable, juste l'affichage
11	**Créer composant CLI** → `ng generate component` → ⚠ ng new = projet entier, pas composant
12	**Préfixe directives structurelles** → `*` (astérisque) → ⚠ Sans *, Angular cherche directive d'attribut
13	**Métadonnées composant** → décorateur `@Component` → ⚠ Constructeur = injection de dépendances
14	**Requêtes HTTP** → `HttpClientModule` → ⚠ HttpModule est DÉPRÉCIÉ depuis Angular 4.3
15	**Opérateur filter()** → Bloque données non conformes → ⚠ filter ne transforme pas, il laisse passer ou bloque
16	**Content Projection** → `<ng-content>` → ⚠ Utile pour composants génériques (Card, Modal)
17	**Hook avant destruction** → `ngOnDestroy` → ⚠ CAPITAL pour unsubscribe (éviter memory leaks)

#	Concept → Réponse → Piège
18	**Event binding click** → `(click)` → ⚠ Pas de "on" devant : (click) pas (onclick)
19	**Router.events type** → Observable NavigationEvent → ⚠ C'est un Observable, donc .subscribe() requis
20	**Accès variable #form** → `@ViewChild()` → ⚠ Disponible à partir de ngAfterViewInit seulement
21	**Fonction NgModule** → Organiser et compiler le code → ⚠ Depuis Angular 14, modules optionnels (Standalone)
22	**Observable dans template** → pipe `async` → ⚠ Auto-subscribe ET auto-unsubscribe (pas de memory leak)
23	**ngFor "let item of list"** → `of` spécifique Angular → ⚠ Ne pas confondre avec `in` (pour les clés JS)
24	**Importer service** → import fichier + injection constructeur → ⚠ Les DEUX sont nécessaires
25	**TS vers JS** → Transpiler → ⚠ Compilateur = vers niveau inférieur, Transpiler = même niveau
26	**Émettre événements** → `@Output()` → ⚠ Nécessite EventEmitter, parenthèses obligatoires
27	**Tree Shaking** → Nettoyer le code mort → ⚠ C'est pourquoi providedIn: 'root' est préférable
28	**Intercepter HTTP** → `HttpInterceptor` → ⚠ Enregistrer avec token HTTP_INTERCEPTORS
29	**Infos URL active** → `ActivatedRoute` → ⚠ Router = ACTION naviguer, ActivatedRoute = ÉTAT actuel
30	**Injection optionnelle** → `@Optional()` → ⚠ Retourne null si introuvable (pas d'erreur)
31	**Build production** → `ng build --configuration production` → ⚠ Bundle plus petit que ng serve
32	**Unité de base Angular** → Le Composant → ⚠ Module = conteneur, Composant = bloc fonctionnel
33	**Versions dépendances** → `package.json` → ⚠ Ne pas modifier manuellement sans npm install
34	**ng-template** → Bloc réutilisable non rendu par défaut → ⚠ Activé par directive (ngIf, ngTemplateOutlet)
35	**Alias chemin tsconfig** → `paths` → ⚠ Coordonner avec baseUrl pour fonctionner
36	**Tests unitaires** → `ng test` → ⚠ Mode watch par défaut (relance auto)
37	**Service singleton** → `providedIn: 'root'` → ⚠ Dans providers composant = instance par composant!
38	**Changer tout FormGroup** → `setValue()` → ⚠ setValue = STRICT (toutes props), patchValue = partiel
39	**Écouter clavier window** → `@HostListener('window:keydown')` → ⚠ Préférable à addEventListener
40	**Opérateur tap()** → Déboguer/Effets de bord → ⚠ tap ne modifie PAS la donnée (use map for that)

📗 SECTION 2 – Questions Examens (60 Questions)

#	Concept → Réponse → Piège
1	**Premier appel après création** → `constructor` (JS/TS) → ⚠ ngOnInit vient APRÈS, c'est le hook Angular
2	**Injection service** → constructor(private ds: DataService) OU inject() → ⚠ Jamais new Service()!
3	**Observable sans subscribe** → Rien ne se passe → ⚠ Les Observables sont "froids" = pas de subscribe = pas de requête
4	**Écouter événements hôte** → `@HostListener` → ⚠ @HostBinding MODIFIE, @HostListener ÉCOUTE
5	**Route param obligatoire** → `:id` (deux-points) → ⚠ Les accolades {id} = Spring/Java, pas Angular
6	**Tests temps réel** → `ng test` (watch par défaut) → ⚠ En CI/CD, utiliser --watch=false
7	**Retour validateur sync** → null (valide) ou objet d'erreur → ⚠ Pas true/false, Angular a besoin de l'objet
8	**Service limité au module** → dans `providers: []` du module → ⚠ root = singleton global, providers = scope limité
9	**switchMap comportement** → Annule l'Observable précédent → ⚠ Dangereux pour opérations d'écriture
10	**Interface pour Pipe** → `PipeTransform` → ⚠ Les pipes doivent être "purs" pour éviter recalculs
11	**Config Angular CLI** → `angular.json` → ⚠ Confusion avec package.json (dépendances npm)
12	**Mode AOT** → Compile pendant le build → ⚠ JIT = navigateur, AOT = avant téléchargement
13	**Désactiver bouton si invalide** → `[disabled]="form.invalid"` → ⚠ (disabled) = Event binding = FAUX
14	**Subject avec valeur initiale** → `BehaviorSubject` → ⚠ Subject classique n'émet rien à l'abonnement
15	**Erreur HTTP + valeur secours** → `catchError()` → ⚠ DOIT retourner un Observable, sinon crash
16	**Boucler sur tableau** → `*ngFor` → ⚠ Utiliser trackBy pour optimiser si liste change souvent
17	**@Input change → hook** → `ngOnChanges` → ⚠ Se déclenche que si RÉFÉRENCE change, pas mutation
18	**Contenu entre balises** → `<ng-content>` → ⚠ Content Projection avec sélecteurs CSS possibles
19	**--base-href /app/** → Définit chemin racine liens → ⚠ Mauvais base-href = erreurs 404 JS/CSS
20	**Retarder émissions** → `delay()` → ⚠ debounceTime = attendre silence, delay = décaler tout
21	**Valeur formulaire réactif** → form.value OU form.getRawValue() → ⚠ .value exclut champs désactivés!
22	**But Zone.js** → Intercepter tâches asynchrones → ⚠ Angular 18+ : apps "Zoneless" possibles

#	Concept → Réponse → Piège
23	**Composant si admin** → Guard canActivate + *ngIf → ⚠ *ngIf seul n'est pas sécurité (URL directe)
24	**Détection changements manuelle** → detectChanges() OU markForCheck() → ⚠ Éviter detectChanges() abusif
25	**Variable CSS hôte** → `@HostBinding` → ⚠ Plus propre que ElementRef
26	**Afficher template par nom** → `ngTemplateOutlet` → ⚠ ng-template définit mais n'affiche pas
27	**Réagir touche Entrée** → `(keydown.enter)` → ⚠ event.keyCode === 13 est obsolète
28	**providedIn: 'any'** → Instance par module lazy → ⚠ Rarement utilisé, préférer 'root'
29	**Rôle du Pipe** → Transformer données pour affichage → ⚠ Pipes = VUE, logique métier = service
30	**Module avec routing** → `ng g m my-mod --routing` → ⚠ Sans --routing, création manuelle requise
31	**Protection XSS** → Angular automatique → ⚠ DomSanitizer pour innerHTML avec données user
32	**Décorateur requis composant** → `@Component` → ⚠ Sans lui, classe = simple objet TS
33	**Parent → Enfant** → `@Input()` → ⚠ Direction : @Input = vers intérieur, @Output = vers extérieur
34	**Hook avant ngOnInit** → `ngOnChanges` → ⚠ constructor n'est pas un hook Angular
35	**Envoyer événements** → `EventEmitter` → ⚠ Optimisé pour interactions avec moteur rendu
36	**Lier propriété HTML** → `[property]` ☐ = Property Binding (+ performant)
37	**AOT signification** → Ahead Of Time → ⚠ Détecte erreurs template au build
38	**Style isolé composant** → styles/styleUrls + ViewEncapsulation.Emulated → ⚠ ViewEncapsulation.None = fuite CSS
39	**Installer Angular CLI** → `npm install -g @angular/cli` → ⚠ -g obligatoire sinon local au projet
40	**polyfills.ts rôle** → Compatibilité IE11/vieux navigateurs → ⚠ Presque vide avec navigateurs modernes
41	**Template vs Reactive Forms** → Logique template vs TS → ⚠ Reactive = recommandé pour complexe/tests
42	**Importer module externe** → dans `imports: []` → ⚠ declarations = composants locaux uniquement
43	**Combiner Observables attendre fin** → `forkJoin` → ⚠ Si UN Observable échoue, tout forkJoin échoue
44	**Validateur async personnalisé** → Retourner Observable → ⚠ Marquer champ comme 'pending' pendant validation
45	**Stockage client** → `localStorage` (API native) → ⚠ Angular n'a pas de service intégré pour ça
46	**trackBy dans ngFor* → Améliorer performances → ⚠ Sans trackBy, Angular recrée tout le DOM

#	Concept → Réponse → Piège
47	**Données sans parent-enfant** → Service avec BehaviorSubject → ⚠ localStorage persiste mais pas réactif
48	**pipe async** → S'abonne auto à Observable → ⚠ Se désabonne auto à destruction (recommandé)
49	**Lazy-load module** → `loadChildren: () => import(...)` → ⚠ Module lazy = son propre injecteur
50	**Accès contenu projeté** → `@ContentChild` → ⚠ @ViewChild = template propre, @ContentChild = projeté
51	**Route par défaut** → `path: ''` → ⚠ path: '**' pour wildcard, pas path: '*'
52	**Guard quitter page** → `canDeactivate` → ⚠ Utile pour formulaires non sauvegardés
53	**Données via URL** → Query params OU Route params OU data → ⚠ Route params obligatoires, query optionnels
54	**resolve dans route** → Pré-charger données → ⚠ Composant ne s'affiche pas tant que resolver pas fini
55	**Créer intercepteur HTTP** → Implémenter `HttpInterceptor` → ⚠ S'applique à TOUTES les requêtes
56	**Ordre hooks** → OnChanges → OnInit → AfterViewInit → ⚠ OnChanges AVANT OnInit si @Input existe
57	**Composant standalone** → `standalone: true` → ⚠ Imports directement dans @Component
58	**Ignorer nouvelles valeurs** → `exhaustMap` → ⚠ Idéal pour éviter doubles soumissions formulaire
59	**Enfants directs template** → `@ViewChildren` → ⚠ @ContentChildren = contenu projeté
60	**ngOnDestroy rôle** → Nettoyer ressources → ⚠ Si pas unsubscribe = memory leaks

📙 SECTION 3 – Screenshots & Analyse Code (11 Questions)

#	Concept → Réponse → Piège
1	**@Input() manquant** → Import de Input depuis @angular/core → ⚠ Sur papier, on oublie import en haut
2	**user null avec ngIf* → Div non rendu, aucune erreur → ⚠ Méthode #1 pour éviter null pointer
3	**Index dans ngFor* → `let i = index` ou `index as i` → ⚠ Jamais `let i of index`
4	**ng-container dans DOM** → Ne crée rien (invisible) → ⚠ Utile pour *ngIf et *ngFor même niveau
5	**Two-Way Binding syntaxe** → `[(ngModel)]` → ⚠ FormsModule requis
6	**user?.name syntaxe** → Safe navigation operator → ⚠ Angular = "Safe navigation", JS = "Optional chaining"
7	**Button dans form** → Type par défaut = submit → ⚠ Ajoutez type="button" pour éviter soumission

#	Concept → Réponse → Piège
8	**Appeler méthode enfant** → `@ViewChild` → ⚠ Référence disponible après ngAfterViewInit
9	**[class.x] vs [ngClass]** → ngClass permet plusieurs classes → ⚠ Pour une seule classe, [class.x] plus lisible
10	**exportAs directive** → Accès via variable de référence → ⚠ C'est ainsi que ngModel exporte FormControl
11	**Accès DOM natif** → ElementRef OU @ViewChild.nativeElement → ⚠ document.getElementById contourne Angular

📕 SECTION 6 – Code Approfondi (50 Questions)

🗺 Routing (10Q)

#	Concept → Réponse → Piège
1	**Route enfant** → `children: [{ path: 'child', component: ... }]` → ⚠ Parent doit avoir
2	**Guard canActivate fonctionnel** → `CanActivateFn` (fonction) OU classe → ⚠ Les deux valides Angular 15+
3	**snapshot vs params Observable** → snapshot = figé, params = réactif → ⚠ Même composant différent ID → snapshot pas mis à jour!
4	**Retour Resolver** → valeur, Promise, ou Observable → ⚠ Route activée APRÈS fin du Resolver
5	**Accès données Resolver** → `route.snapshot.data['key']` ou Observable → ⚠ Utiliser la même clé que dans resolve: {}
6	**Formulaire non sauvé** → `canDeactivate` → ⚠ Peut retourner confirm() pour demander confirmation
7	**Données statiques route** → `data: { title: 'Accueil' }` → ⚠ Accessible via route.snapshot.data
8	**Route 404 wildcard** → `path: '**'` → ⚠ DOIT être en dernier sinon capture tout
9	**Navigation query params** → `router.navigate(['/x'], { queryParams: { id: 1 } })` → ⚠ navigate(['/x?id=1']) ne fonctionne pas
10	**Précharger tous lazy modules** → `preloadingStrategy: PreloadAllModules` → ⚠ Possible stratégie personnalisée

🔗 Data Binding (10Q)

#	Concept → Réponse → Piège
11	**Valeur par défaut @Input** → `@Input() value = 'default'` → ⚠ String dans @Input() = alias, pas défaut

#	Concept → Réponse → Piège
12	**@Input obligatoire** → `@Input({ required: true })` (Angular 16+) → ⚠ ! seul = TypeScript, pas runtime
13	**EventEmitter typé** → `new EventEmitter<string>()` → ⚠ TypeScript vérifie le type émis
14	**@ViewChild vs @ContentChild** → View = template propre, Content = projeté → ⚠ Dispo dans hooks différents
15	**Plusieurs éléments même sélecteur** → `@ViewChildren` → QueryList → ⚠ QueryList.changes = Observable
16	**@ViewChild disponible quand** → `ngAfterViewInit` → ⚠ Dans ngOnInit = undefined!
17	**static: true ViewChild** → Disponible dans ngOnInit → ⚠ Seulement si élément PAS dans *ngIf/*ngFor
18	**Two-way binding custom** → @Input() prop + @Output() propChange → ⚠ Nom Output = Input + "Change" OBLIGATOIRE
19	**Transformer @Input** → Setter : `@Input() set value(v)` → ⚠ Ajouter getter pour lire valeur transformée
20	**Passer template en @Input** → `@Input() template: TemplateRef<any>` → ⚠ Utiliser ngTemplateOutlet pour afficher

📡 RxJS & HTTP (10Q)

#	Concept → Réponse → Piège
21	**Combiner, émettre dès qu'un émet** → `merge()` → ⚠ combineLatest attend que TOUS aient émis au moins 1×
22	**Retry 3 fois** → `retry(3)` ou `retryWhen` → ⚠ retry = immédiat, retryWhen = délai possible
23	**Valeur par défaut si vide** → `defaultIfEmpty('val')` → ⚠ startWith émet TOUJOURS, même si Observable émet après
24	**Annuler après 5s** → `timeout(5000)` ou `takeUntil(timer(5000))` → ⚠ timeout = erreur, takeUntil = silencieux
25	**Partager Observable** → `share()` ou `shareReplay(1)` → ⚠ Sans partage, chaque subscribe = nouvelle requête!
26	**404 → tableau vide** → `catchError(err => err.status === 404 ? of([]) : throwError(err))` → ⚠ Vérifier status pour comportement spécifique
27	**Debounce 300ms** → `debounceTime(300)` → ⚠ debounce() prend Observable, pas nombre
28	**Throttle 1/sec** → `throttleTime(1000)` ou `sampleTime(1000)` → ⚠ throttle = première, sample = dernière
29	**Code après émission ET erreur** → `finalize()` → ⚠ Idéal pour cacher spinner

#	Concept → Réponse → Piège
30	**Promise → Observable** → `from(promise)` → ⚠ of(promise) émet la Promise comme objet!

📝 Forms (10Q)

#	Concept → Réponse → Piège
31	**Validateur multi-champs** → Appliquer sur FormGroup → ⚠ Ex: comparaison password/confirmPassword
32	**Afficher erreur custom** → `control.hasError('myError')` ou `.errors?.myError` → ⚠ Toutes valides
33	**Marquer tous touched** → `form.markAllAsTouched()` → ⚠ Utile pour afficher toutes erreurs à la soumission
34	**Ajouter validateur dynamique** → `setValidators([])` ou `addValidator()` → ⚠ Appeler updateValueAndValidity() après
35	**FormArray de FormGroups** → `fb.array([fb.group({ name: '' })])` → ⚠ FormBuilder plus lisible pour complexe
36	**Accès contrôle FormArray** → `.at(index)` ou `.controls[index]` → ⚠ Caster en FormGroup si nécessaire
37	**Écouter changements valeur** → `control.valueChanges.subscribe()` → ⚠ Ne pas oublier unsubscribe!
38	**Désactiver validation temp** → `clearValidators()` ou `setValidators(null)` → ⚠ disable() garde validateurs
39	**Validateur async email unique** → Retourner Observable/Promise → ⚠ Utiliser debounce pour éviter spam serveur
40	**Réinit formulaire** → `form.reset()` ou `form.reset(initialValues)` → ⚠ reset() réinitialise aussi touched/dirty

🏗 Architecture (10Q)

#	Concept → Réponse → Piège
41	**InjectionToken config** → `new InjectionToken<Config>('config')` → ⚠ Utile pour injecter valeurs, pas classes
42	**Fournir valeur InjectionToken** → `{ provide: TOKEN, useValue: 'val' }` → ⚠ useFactory pour calcul dynamique
43	**useClass/useValue/useFactory/useExisting** → Classe/Constante/Fonction/Alias → ⚠ useExisting = alias vers autre provider
44	**Service scope composant** → `@Component({ providers: [Service] })` → ⚠ Chaque instance composant = son instance service

#	Concept → Réponse → Piège
45	**Injecteur parent** → `@SkipSelf()` → ⚠ Ignore injecteur local, cherche chez parents
46	**Limiter recherche service** → `@Host()` → ⚠ @Self() = uniquement composant actuel
47	**Injection optionnelle** → `@Optional()` → ⚠ Retourne null, pas d'erreur. Vérifier avant utilisation!
48	**Ré-exporter module** → `imports: [M], exports: [M]` → ⚠ Il faut importer ET exporter
49	**Module exposer certains composants** → declarations = tous, exports = publics → ⚠ Seuls exports sont accessibles
50	**Éviter imports circulaires** → SharedModule / forwardRef / lazy-loading → ⚠ Éviter dépendances mutuelles dès le design

📒 SECTION 7 – Focus Examen (26 Questions)

🗺 Navigation Déclarative (5Q)

#	Concept → Réponse → Piège
1	**Où injecter composant routé** → `<router-outlet>` → ⚠ Sans lui, route matche mais rien ne s'affiche
2	**Naviguer sans reload** → `routerLink` → ⚠ href force rechargement complet (pas SPA)
3	**routerLink avec id** → `[routerLink]="['/users', user.id]"` → ⚠ :id = placeholder routes, pas routerLink
4	**routerLinkActive** → Ajoute classe CSS si route active → ⚠ Match partiel par défaut (prefix)
5	**Match exact routerLinkActive** → `[routerLinkActiveOptions]="{ exact: true }"` → ⚠ Sans exact, /users active aussi /users/123

📡 HTTP Fondamentaux (6Q)

#	Concept → Réponse → Piège
6	**Récupérer données** → `http.get()` → ⚠ Méthodes = verbes HTTP
7	**Créer ressource** → `POST` → ⚠ PUT remplace, POST crée
8	**Remplacer ressource entière** → `PUT` → ⚠ Confusion PUT (total) vs PATCH (partiel)
9	**Supprimer ressource** → `DELETE` → ⚠ POST peut supprimer côté backend mais pas convention REST
10	**Ajouter header Auth** → `{ headers: new HttpHeaders({ Authorization: 'Bearer token' }) }` → ⚠ HttpHeaders IMMUTABLE
11	**Récupérer status/headers** → `{ observe: 'response' }` → ⚠ Par défaut, HttpClient retourne juste body

🔧 Pipes (5Q)

#	Concept → Réponse → Piège
12	**Pipe par défaut** → Pur (pure: true) → ⚠ Pipe pur = pas réévalué chaque cycle
13	**Pipe pur se réévalue quand** → Référence/valeur primitive change → ⚠ Muter objet/array ne change pas référence
14	**array.push() pipe pur** → Pas de mise à jour → ⚠ push garde même référence. Utiliser [...items, x]
15	**Pipe impur** → `@Pipe({ pure: false })` → ⚠ Réévalué CHAQUE cycle de détection
16	**Risque pipe impur** → Problèmes performance → ⚠ Exécuté à chaque cycle CD = coûteux

⚡ Composants Dynamiques (5Q)

#	Concept → Réponse → Piège
17	**Créer composant dynamique** → `viewContainerRef.createComponent(MyComponent)` → ⚠ ComponentFactoryResolver = ancienne approche
18	**Composant dynamique template** → `ngComponentOutlet` → ⚠ ngTemplateOutlet = templates, pas composants
19	**Récupérer ViewContainerRef** → `@ViewChild('host', { read: ViewContainerRef })` → ⚠ Sans read, retourne ElementRef
20	**Passer @Input composant dynamique** → `componentRef.instance.title = 'Hello'` → ⚠ Avec OnPush, detectChanges() peut être nécessaire
21	**Vider conteneur dynamique** → `viewContainerRef.clear()` → ⚠ .destroy() = une instance spécifique

🏗 Directives Structurelles (5Q)

#	Concept → Réponse → Piège
22	**Créer directive structurelle** → `TemplateRef + ViewContainerRef` → ⚠ Directives structurelles = vues, pas DOM direct
23	**Signification astérisque (*)** → Sugar pour `<ng-template>` → ⚠ *appX = <ng-template [appX]>...
24	**Créer vue depuis TemplateRef** → `viewContainerRef.createEmbeddedView(templateRef)` → ⚠ createComponent = composant, pas template
25	**Rôle ViewContainerRef** → Insérer/retirer vues DOM → ⚠ Clé pour afficher/masquer conditionnel
26	*@Input pour appUnless* → `@Input() appUnless` → ⚠ Nom doit matcher sélecteur directive

🎯 MÉMO ULTIME – Les 10 Pièges les Plus Fréquents

1. **Observable sans subscribe() = requête JAMAIS exécutée**
2. **@Service() N'EXISTE PAS** (c'est @Injectable())

3. **pristine ≠ touched** (modification vs visite)
4. **FormsModule pour ngModel, ReactiveFormsModule pour FormGroup**
5. **Route param = `:id`, pas `{id}`** (: pas accolades)
6. **setValue() = TOUS les champs, patchValue() = partiel**
7. **@ViewChild dispo dans ngAfterViewInit, PAS ngOnInit**
8. **switch/merge/exhaust/concatMap** → Savoir lequel annule/cumule/ignore/séquence
9. **Pipe pur = ne se met pas à jour si mutation** (utiliser spread [...])
10. **async pipe = auto-subscribe ET auto-unsubscribe** (pas de memory leak)

📚 *Fichier généré pour révision rapide. Chaque ligne = 1 question mémorisée !*