🏆 Angular Code Holy Grail - Examen

> Le minimum VITAL de code à connaître par cœur

1️⃣ COMPOSANT DE BASE

```
@Component({
  selector: 'app-example',
  templateUrl: './example.component.html',
  standalone: true,
  imports: [CommonModule, FormsModule]
})
export class ExampleComponent implements OnInit, OnDestroy {
  @Input() data!: string;           // Parent → Enfant
  @Output() send = new EventEmitter<string>(); // Enfant → Parent

  ngOnInit() { /* init */ }
  ngOnDestroy() { /* cleanup */ }
}
```

2️⃣ SERVICE + INJECTION

```
@Injectable({ providedIn: 'root' })  // Singleton global
export class DataService {
  private data$ = new BehaviorSubject<string[]>([]);

  constructor(private http: HttpClient) {}

  getItems(): Observable<Item[]> {
    return this.http.get<Item[]>('/api/items');
  }
}
```

3️⃣ ROUTING

```
// app.routes.ts
export const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'user/:id', component: UserComponent },  // :id = paramètre
  { path: 'admin', loadChildren: () => import('./admin/admin.routes') },
// Lazy
  { path: '**', redirectTo: '' }  // Fallback
```

```
];

// Lire paramètre dans composant
constructor(private route: ActivatedRoute) {}
ngOnInit() {
  this.route.params.subscribe(p => this.id = p['id']);
  // OU snapshot (non-réactif)
  this.id = this.route.snapshot.params['id'];
}
```

4️⃣ FORMULAIRE RÉACTIF

```
// Dans le composant
form = new FormGroup({
  email: new FormControl('', [Validators.required, Validators.email]),
  password: new FormControl('', Validators.minLength(6))
});

onSubmit() {
  if (this.form.valid) {
    console.log(this.form.value);
  }
}
```

```
<!-- Template -->
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <input formControlName="email">
  <span *ngIf="form.get('email')?.errors?.['required']">Requis</span>
  <button [disabled]="form.invalid">Envoyer</button>
</form>
```

5️⃣ DIRECTIVES ESSENTIELLES

```
<!-- Structurelles -->
<div *ngIf="condition; else other">Visible</div>
<ng-template #other>Alternative</ng-template>

<li *ngFor="let item of items; let i = index; trackBy: trackById">
  {{i}}: {{item.name}}
</li>

<!-- Attribut -->
<div [ngClass]="{'active': isActive, 'error': hasError}"></div>
<div [ngStyle]="{'color': textColor}"></div>
```

```
<!-- Two-way binding -->
<input [(ngModel)]="name">
```

6️⃣ HTTP + INTERCEPTOR

```
// Service HTTP
getData(): Observable<Data[]> {
  return this.http.get<Data[]>('/api/data').pipe(
    catchError(err => {
      console.error(err);
      return of([]);  // Fallback
    })
  );
}

// INTERCEPTOR (ajouter token)
@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler) {
    const authReq = req.clone({
      setHeaders: { Authorization: `Bearer ${token}` }
    });
    return next.handle(authReq);
  }
}
```

7️⃣ PIPE PERSONNALISÉ

```
@Pipe({ name: 'truncate', standalone: true })
export class TruncatePipe implements PipeTransform {
  transform(value: string, limit = 20): string {
    return value.length > limit ? value.slice(0, limit) + '...' : value;
  }
}
// Usage: {{ text | truncate:30 }}
```

8️⃣ GUARD (CanActivate)

```
export const authGuard: CanActivateFn = (route, state) => {
  const auth = inject(AuthService);
  const router = inject(Router);
```

```
  if (auth.isLoggedIn()) return true;
  return router.createUrlTree(['/login']);
};

// Dans routes:
{ path: 'admin', component: AdminComponent, canActivate: [authGuard] }
```

9️⃣ RXJS ESSENTIEL

```
// subscribe + unsubscribe
private destroy$ = new Subject<void>();

ngOnInit() {
  this.data$.pipe(
    takeUntil(this.destroy$)
  ).subscribe(d => this.data = d);
}

ngOnDestroy() {
  this.destroy$.next();
  this.destroy$.complete();
}

// OU avec async pipe (auto-unsubscribe)
// {{ data$ | async }}
```

🔑 MÉMO RAPIDE

Concept	Code clé
Input	`@Input() prop: T`
Output	`@Output() evt = new EventEmitter<T>()`
Service singleton	`@Injectable({providedIn:'root'})`
Route param	`route.params.subscribe()` ou `snapshot.params['x']`
Form control	`new FormControl('', Validators.required)`
HTTP GET	`http.get<T>(url).subscribe()`
Pipe	`{{ val | pipe:arg }}`
Guard	`canActivate: [guardFn]`
