Bienvenue

- Moderniser une application métier Java/Angular
- Structurer une librairie de composants réutilisable
- Garantir qualité, accessibilité et time-to-market

Objectifs

1. Mettre en place une librairie Angular modulaire.
2. Factoriser les formulaires via typed forms + CVA.
3. Définir conventions & tokens pour un design system durable.
4. Sécuriser la qualité (tests ciblés, CI) et la diffusion.

Agenda (4h)

Créneau	Sujet	Livrable
10’	Kick-off & démo cible	Vision partagée
40’	Lib Angular & architecture	Module partagé
50’	Composants réutilisables	Kit boutons/inputs
50’	Typed forms & CVA	Form field typé
40’	Design system & tokens	Thème + tokens
30’	Tests & a11y	Specs unitaires
20’	CI & publish local	Pipeline de base

Kick-off (10’)

- **Pourquoi** : réduire dette front, accélérer nouveaux écrans.
- **Comment** : lib Angular isolée (OnPush), process de revue + CI.
- **Résultat attendu** : starter kit livrable, conventions validées.

> *Démo cible : composant `app-field-select` accessible + story Storybook.*

Création de la librairie (40’)

Étapes clés

- `ng generate library shared-ui` (architecture nx-like).
- Configurer `tsconfig` pour les paths (`@shared/ui`).
- Activer `ChangeDetectionStrategy.OnPush` par défaut.

```ts
// projects/shared-ui/src/lib/button/button.component.ts
@Component({
  selector: "shared-button",
  changeDetection: ChangeDetectionStrategy.OnPush,
  templateUrl: "./button.component.html",
})
export class ButtonComponent {
  @Input({ required: true }) label!: string;
  @Input() kind: "primary" | "ghost" = "primary";
}
```

Exercice 01 · Bases de la lib

- Objectif : générer la lib `shared-ui`, exposer un premier composant.
- Préflight : lint ok, build de la lib ok (`ng build shared-ui`).
- Pitfalls : oublier `ng-packagr`, exposer modules publics.

> *Pair programming recommandé pour aligner conventions.*

Composants réutilisables (50’)

Focus : `shared-button`, `shared-input`, `shared-form-field`.

- Inputs typés (`@Input({ transform })`).
- Slots via `ng-content` + `cdkTrapFocus` pour modales.
- Gestion des états (loading, disabled) + tokens de thème.

```html
<label shared-form-field [state]="state">
  <span label>Libellé</span>
  <input shared-input formControlName="email" />
  <p hint>Email pro requis</p>
</label>
```

Exercice 02 · Formulaires typés

- Créer un `FormGroup` typé via `FormBuilder.nonNullable`.
- Implémenter un `ControlValueAccessor` pour `shared-select`.
- Couvrir validations + messages d’erreur dynamiques.

Critères de réussite :

- TypeScript interdit les `any`.
- `ARIA` (`aria-invalid`, `role="alert"`) gérés automatiquement.

Typed forms & CVA (50’)

- Pattern `ControlValueAccessor` + `provideNgValueAccessor`.
- Générer un wrapper `shared-form-field` => label, hint, errors.
- Support `FormControl<string | null>` & `FormControl<Date | null>`.

Tips :

- Utiliser `signal` ou `computed` pour l'état local si Angular 17.
- Tests unitaires sur `writeValue`, `registerOnChange`.

Design system (40')

- Tokens (spacing, radius, color) exposés via SCSS/CSS vars.
- Variants (primary, ghost, destructive) normalisés.
- i18n : texte par défaut externalisé (`@ngx-translate` ou équivalent).

```
:root {
  --shared-spacing-4: 1rem;
  --shared-color-primary: var(--brand-emerald-500);
}
```

Checklist rapide : naming, responsive, docs, capture Storybook.

Exercice 03 · Design system

1. Ajouter un jeu de tokens pour le thème dark.
2. Documenter les variants dans Storybook (Controls + Docs).
3. Exporter une `DesignGuidelines.md` (10 bullet points max).

Résultat attendu : une story "Guidelines" partagée avec le métier.

Tests & accessibilité (30')

- Tests unitaires ciblés (`TestBed`, `spectator`).
- Axe-core sur Storybook (`@storybook/addon-a11y`).
- Matrice WCAG AA : contrastes, focus ring, tab order.

```
it('render aria-invalid when control invalid', () => {
  control.setValue('');
  control.markAsTouched();
  fixture.detectChanges();
  expect(fieldElement.getAttribute('aria-invalid')).toBe('true');
});
```

Exercice 04 · Tests ciblés

- Vérifier `aria-describedby`, `tabindex`, gestion des erreurs.
- Ajouter tests screenshot si possible (Chromatic, Playwright).

- Documenter pitfalls dans `solutions/04-testing.md`.

CI & diffusion (20’)

- Pipeline npm scripts : `lint`, `test`, `build`, `publish:prerelease`.
- Registry local (Verdaccio ou GitHub Packages) pour diffusion interne.
- Gates : lint/test obligatoires, storybook build (a11y).

Exercice 05 · CI intégration

1. Ajouter job `lint` + `test` + `build` sur chaque PR.
2. Publier un package `0.0.0-pr.<sha>` sur registry local.
3. Archiver la checklist CI signée par le tech lead.

Synthèse & next steps

- Librairie partagée opérationnelle.
- Formulaires typés + CVA prêts pour production.
- Design system aligné (tokens + variants + stories).
- Pipeline CI activé, process d’industrialisation amorcé.

Prochaines étapes : audit complet DS, intégration stricte a11y, documentation métier.

Questions / Feedback

- Quelles pages veulent être migrées en priorité ?
- Quels KPI suivre (lead time, bugs, satisfaction UX) ?
- Prochaine session : audit design system avancé ou workshop CI/CD ?