Guía de estilos para Angular (basada en Google)

1. Introducción

Esta guía define convenciones y buenas prácticas para proyectos Angular, inspirada en el Style Guide de Google. Está pensada para garantizar legibilidad, mantenibilidad y calidad de software.

---

2. Convenciones de nombres

- **Clases, interfaces y tipos**: `UpperCamelCase` (p.ej., `UserProfileComponent`, `AuthService`).
- **Componentes, directivas y pipes**: sufijo explícito (`Component`, `Directive`, `Pipe`).
- **Archivos**: `kebab-case` (p.ej., `user-profile.component.ts`).
- **Variables y métodos**: `lowerCamelCase` (p.ej., `getUserData()`, `userList`).
- **Constantes**: `UPPER_SNAKE_CASE` (p.ej., `API_ENDPOINT`).

---

3. Estructura del proyecto

Organizar el código por características (feature-based):

```
src/app/
  core/               # Servicios singleton, guardas, interceptors
  shared/             # Componentes, pipes y directivas reutilizables
  feature-name/       # Carpetas por cada módulo de funcionalidad
    components/
    services/
    models/
    feature-name.module.ts
    feature-name-routing.module.ts
```

- **core/**: sólo código que se importa una vez en `AppModule`.
- **shared/**: módulos compartidos que se importan en múltiples features.

---

4. Componentes

- **Nombre**: `<Feature><Element>Component` (p.ej., `LoginFormComponent`).
- **Selector**: prefijo `app-` y `kebab-case` (p.ej., `app-login-form`).
- **Ficheros**:
  - `.component.ts`, `.component.html`, `.component.scss`, `.component.spec.ts`.
- **Encapsulación**: usar `ViewEncapsulation.Emulated` (por defecto).

---

5. Módulos

- **Naming**: `<Feature>Module` (p.ej., `AuthModule`).
- **Barrel files (`index.ts`)**: exportar entidades públicas.
- **Feature Modules**: deben declarar y exportar sólo lo necesario.

---

6. Servicios

- **Naming**: `<Feature>Service` (p.ej., `AuthService`).

- **Decorador**: `@Injectable({ providedIn: 'root' })` para singleton.
- **Responsabilidad única**: cada servicio gestiona una única tarea o recurso.

---

7. Directivas y Pipes

- **Sufijos**: `Directive` y `Pipe` (p.ej., `HighlightDirective`, `DateFormatPipe`).
- **Uso**: directivas para manipular DOM, pipes para formateo de datos en plantillas.

---

8. Templates y Bindings

- **Interpolación** (`{{ }}`) para texto estático.
- **Property binding** (`[attr]`) para propiedades de DOM.
- **Event binding** (`(event)`) para eventos del usuario.
- **Two-way binding** (`[(ngModel)]`) con moderación.
- **Estructurales**: usar `*ngIf`, `*ngFor` con `trackBy`.

---

9. Estilos y CSS/SCSS

- **Archivos SCSS**: preferir variables, mixins y BEM.
- **Scoped styles**: dentro del componente.
- **Variables globales**: definir en `styles.scss`.

---

10. Linting y Formateo

- **ESLint**: configurar reglas específicas de Angular.
- **Prettier**: para consistencia en formateo.
- **Hooks de git**: `lint-staged` antes de commits.

---

11. Comentarios y Documentación

- **TSDoc/JSDoc**: documentar funciones públicas y componentes.
- **Comentarios inline**: sólo para lógica compleja.

---

12. Testing

- **Especificaciones**: archivo `.spec.ts` por cada componente/servicio.
- **Estructura**: `describe` para agrupar, `it` para casos de prueba.
- **Arrange-Act-Assert**: seguir este patrón.

---

13. Buenas prácticas de rendimiento

- **ChangeDetectionStrategy.OnPush** en componentes puros.
- **Lazy loading** de módulos.
- **Unsubscribe** en observables para evitar memory leaks.
- **trackBy** en `*ngFor` para optimizar render.

---

14. Seguridad

- **Sanitización**: usar `DomSanitizer` para HTML dinámico.
- **Protección**: implementar guards (`CanActivate` , `CanLoad`).

---

15. Referencias

Este documento se base en el Style Guide de Google para proyectos front-end