ANGULAR
ARCHITECTS

# Angular Styling

Alexander Thalhammer | @LX_T

# Angular Styling

- [ngClass] vs [class.class-name]
- inject(DOCUMENT)
- (S)CSS Architecture
- View Encapsulation
- NG 17 View Transitions
- Component Frameworks in NG
- Design Systems for NG

Agenda

ANGULAR
ARCHITECTS

# [ngClass] vs [class.class-name]

– which one do you like better

```
<tr *ngFor="let f of flightsSignal()" [ngClass]="{ active: f.id === selectedFlight.id }">
  …
</tr>
```

```
<tr *ngFor="let f of flightsSignal()" [class.active]="f.id === selectedFlight.id">
  …
</tr>
```

# inject(DOCUMENT)

- use DOCUMENT token
  - when accessing document via Angular
  - to support browser APIs and SSR

```
private readonly document = inject(DOCUMENT);
```

```
// get body
const body = this.document.getElementsByTagName('body')[0];

// scroll to top
this.document.documentElement.scrollTop = 0;
```

## (S)CSS Architecture

- Use Component-Based SCSS
  - Avoid global styles – except when they make sense
- Consistent Naming, Linting & Formatting
  - Prettier & Stylelint
- Documentation and Comments
  - complex CSS rules, hacks or workarounds
- Third-Party Stylesheets & Component Frameworks
  - mindful of their impact on your styling & performance

ANGULAR
ARCHITECTS

# View Encapsulation

- **Emulated**
  - Usage: Styles scoped to the component via [attribute]
  - Pros: Prevents style leakage, access to custom props
  - Cons: Pollutes HTML
- ShadowDom
  - Usage: Uses browser's native shadow DOM
  - Pros: Improved performance, accurate scoping
  - Cons: No global styles, no custom props
- None
  - Usage: Disables encapsulation, styles become global
  - Pros: For global components, integrating third-party libs
  - Cons: Pollutes CSS

# Angular 17 View Transitions

```
export const appConfig: ApplicationConfig = {
  providers: [
    provideRouter(
      routes,
      withViewTransitions() // the magic
    ),
  ]
};
```

# Angular 17 View Transitions Customization

```
@keyframes fade-in {
  from { opacity: 0 }
}
@keyframes fade-out {
  to { opacity: 0 }
}
@keyframes slide-from-right {
  from { transform: translateX(30px) }
}
@keyframes slide-to-left {
  to { transform: translateX(-30px) }
}
```

```
::view-transition-old(root) {
  animation:
    90ms cubic-bezier(0.4, 0, 1, 1) both fade-out,
    300ms cubic-bezier(0.4, 0, 0.2, 1) both slide-to-left;
}

::view-transition-new(root) {
  animation:
    210ms cubic-bezier(0, 0, 0.2, 1) 90ms both fade-in,
    300ms cubic-bezier(0.4, 0, 0.2, 1) both slide-from-right;
}
```

## Component Frameworks

- [Angular Material]
- [PrimeNG]
- [NG-ZORRO]
- [Clarity Design]
- [Canopy]

ANGULAR
ARCHITECTS

## Component Frameworks Customization

– a difficult endeavour

– limit customization

– depending on framework

– maybe a own Design System is a better fit

# Angular Material Customization

- theming is supported
  - https://material.angular.io/guide/theming

- custom styling for components
  - by using global SCSS & or component styling (Encapsulation!) https://material.angular.io/guide/customizing-component-styles
  - or extend Angular Material's components
  - make sure to integrate with Material SCSS variables

- 3rd party
  - explore libraries and tools that complement Angular Material

# Design Systems for NG

- Build your own Component Framework (like Canopy)
- Get some inspiration from the others (previous slide)
- Best Practices
  - Tailwind CSS (!)
  - Semantic HTML elements (button, input, ...) &
  - Material CDK for enhanced accessibility
  - ViewEncapsulation.None
  - Storybook for documentation/testing
  - Standalone Components &
  - ChangeDetetctionStrategy.OnPush for performance

## Best Practices

- Use Prettier!
- Use [class.class-name]
  - when dynamic use inline styles [style.property]=
- Put your styles locally into your component.scss
- Organize your global styles with partials
- Use emulated encapsulation as default
- Inject the DOCUMENT token
- Avoid using classes of component frameworks

Tips

## What else is !important?

What else?

ANGULAR ARCHITECTS

# Angular Styling

- [ngClass] vs [class.class-name]
- inject(DOCUMENT)
- (S)CSS Architecture
- View Encapsulation
- Angular 17 View Transitions
- Component Frameworks
- Design Systems

**Summary**

ANGULAR ARCHITECTS

## Questions?

ANGULAR
ARCHITECTS