PrimeNG Migration: Version 14 → Version 15

Overview

PrimeNG 15 builds upon v14 with stabilization of PrimeFlex integration, updates for Angular 15 compatibility, and introduces theme API enhancements. No CSS prefix changes occur, but new theming config options and updated DOM structures for overlays are introduced.

1. Dependency Updates

- Update dependencies in package.json: • "primeng": "^15.0.0" • "primeicons": "^5.0.0" • "primeflex": "^3.2.0"

2. Theming and Theme API

- PrimeNG v15 introduces the `Theme` API for dynamic runtime theme switching via `PrimeNGConfig`. Use `PrimeNGConfig.ripple = true` and `PrimeNGConfig.set('theme', 'lara-light-indigo')` at app initialization. - Legacy CSS variable support remains, but extended variables for new components are added (e.g., --p-overlay-zIndex).

3. Overlay and Dialog Structural Updates

- Overlay components (Toast, Dialog) now include additional wrapper `` for portal management with CSS classes: • `.p-overlay-portal` • `.p-overlay-content` - Verify custom CSS overrides targeting overlays adapt to the new wrapper structure or target new classes.

4. New Components & Enhancements

- Added `AvatarGroup` component for grouping avatars with overflow handling. - Enhanced `Timeline` with lazy loading and virtualization support. - Improved `PickList` accessibility by adding ARIA attributes and keyboard navigation fixes.

5. Migration Steps for v15

1. Update package.json versions and run `npm install`. 2. Import any new modules (e.g., AvatarGroupModule). 3. Update theming initialization code in `app.module.ts` or main. 4. Adjust custom CSS for overlay portal structure. 5. Run unit/E2E tests, focusing on overlay and portal-heavy components. 6. QA new components (AvatarGroup, Timeline) and overall theming.

Angular Upgrade Guide: v18 → v20

Section 1: Upgrade v18 → v19

Angular v19 brings incremental improvements for signals, hydration, and enhances the standalone component model. Prerequisites: - TypeScript 5.6+ - Node.js 18 or 20 Upgrade Steps: 1. Run `ng update @angular/core@19 @angular/cli@19` 2. Migrate any remaining ViewEngine libraries to Ivy-only. 3. Enable

hydration in `angular.json` build options: ```json "server": { "options": { "bundleDependencies": "all", "enableIvy": true, "experimentalHydration": true } } ``` 4. Update standalone component declarations to leverage latest API enhancements. 5. Run tests in SSR mode and validate hydration correctness. 6. Verify e2e tests under production SSR build.

Section 2: Upgrade v19 → v20

Angular v20 stabilizes the signals API and introduces localize enhancements, improved devtools, and updated CLI defaults. Prerequisites: - TypeScript 5.7+ - Node.js 20 Upgrade Steps: 1. Run `ng update @angular/core@20 @angular/cli@20` 2. Enable the experimental `useSignals` option in `tsconfig.json` if using signals: ```json "angularCompilerOptions": { "enableIvy": true, "strictInjectionParameters": true, "useSignals": true } ``` 3. Integrate new `@angular/localize` hydration support for applications requiring i18n – run: `ng add @angular/localize` 4. Review updated CLI defaults: esbuild for all builders, persistent build cache enabled by default. 5. Execute comprehensive test suite and perform performance benchmarking. 6. Conduct QA focusing on i18n flows and devtools integration.