PrimeNG Migration: Version 9 → Version 10

Overview

PrimeNG 10 marks a major overhaul under the “PrimeOne” design system. Key structural changes include renaming all CSS class prefixes from ui- to p-, switching change detection to OnPush, updating PrimeIcons to v4.0, and integrating the PrimeFlex utility CSS library for layouts. This migration guide covers everything needed to upgrade a large Angular application from PrimeNG v9.1.3 to PrimeNG v10.x.

1. Class Prefix Renaming (ui- → p-)

All CSS classes used by PrimeNG components in v9 that started with ui- are renamed to p- in v10. This affects every component’s root, state, and utility classes. For example:

Examples:
- .ui-button → .p-button
- .ui-state-disabled → .p-disabled
- .ui-dialog → .p-dialog, .ui-dialog-titlebar → .p-dialog-header
- .ui-datatable → .p-datatable, .ui-state-highlight → .p-highlight

2. Theming and CSS Variables

PrimeNG v10 introduces CSS variables for theming (e.g., --p-primary-color). Legacy SASS-based themes from v9 ($primaryColor) are still supported if you import the old SASS themes, but you are encouraged to migrate to CSS variable–based themes for runtime theming and dark mode support. Override variables in a global stylesheet:

Example:
```
:root {
--p-primary-color: #007ad9;
--p-primary-text-color: #ffffff;
}
```

3. Dependency Updates

In package.json, bump dependencies to match PrimeNG 10 requirements:

Example:
```
"primeng": "^10.0.0",
"primeicons": "^4.0.0",
"primeflex": "^2.0.0"
```

4. Module Imports

Ensure your Angular modules import the correct PrimeNG modules. Most module names remain unchanged, but verify each import resolves. E.g.:

Example:
```
import { ButtonModule } from 'primeng/button';
import { DialogModule } from 'primeng/dialog';
```

5. Component Structure Changes

Some components had minor DOM structure changes, particularly around content projection slots. The <p-header> and <p-footer> tags were deprecated in favor of <ng-template pTemplate>:

Before:
<p-dialog>
<p-header>Title</p-header>
</p-dialog>
After:
<p-dialog>
<ng-template pTemplate="header">Title</ng-template>
</p-dialog>

6. Removal of Legacy Grid Classes

PrimeNG v9 included basic grid CSS (.ui-g, .ui-g-12). In v10, these were removed in favor of PrimeFlex (.p-grid, .p-col-12). Install and include PrimeFlex CSS:

Command:
npm install primeflex@^2.0.0
angular.json:
"styles": [
"node_modules/primeflex/primeflex.css",
"src/styles.scss"
]

7. Migration Pitfalls & QA

- Update test selectors (e.g., E2E tests using .ui- classes).
- Use visual regression testing to catch styling issues.
- Cross-browser testing: ensure CSS variables support.
- Accessibility: verify ARIA attributes in Dialog/Overlay.
- Premium templates: update their CSS overrides similarly.

8. Step-by-Step Transition Plan

1. Create migration branch.
2. Update dependencies and run npm install.
3. Update module imports.
4. Bulk rename CSS classes (ui- → p-).
5. Update SCSS overrides.
6. Migrate content projection.
7. Include PrimeFlex, adjust grid classes.
8. Run app, fix template errors.
9. Execute tests, update selectors.
10. QA and deploy.

11.0.0

Breaking Changes

- PrimeIcons should be upgraded to 4.1.0 as components like table utilize new icons in the library.

- VirtualScroller implementation of Table is reimplemented and `clearCache` function is removed as caching is left to the page author instead.

- locale property functionality of Calendar is removed in favor of the new global i18n API.

- `filterMode` of Listbox is renamed to filterMatchMode for consistency.

- Pass the option as a template variable in Select Components without wrapping to SelectItem.

Before

```

{{option.value.propName}}

```

After

```

{{option.propName}}

```

Notes

- Yes-no button order in confirm dialog is changed to align it with other Prime UI libraries; it can be reversed with CSS (e.g., `flex-order`).

- PrimeIcons should be upgraded to 4.1.0 as components like table utilize new icons in the library.

10.0.0

PrimeNG 10 is the most important version in the history of the project. A new design system called PrimeOne is integrated, ChangeDetection strategy is changed to OnPush, PrimeIcons have been redesigned for 4.0, PrimeFlex CSS utilities have been updated with new helpers and more. As a result, although prior versions were a drop-in replacement, PrimeNG team has decided to include all major changes to v10 so that future versions can offer a seamless updating experience.

Breaking Changes

- Migration to PrimeOne Design renames style classes; see the PrimeOne Migration page for details.

- PrimeIcons 4.0 should be used with PrimeNG 10 themes; older versions of PrimeIcons are not supported as they use new canvas sizes.

- Modal option removed from Toast.

- `primeng/primeng` were deprecated in v5 and removed in v10.

- Grid CSS was deprecated in 2018 and removed in v10 in favor of PrimeFlex.

- Nova themes have been renamed:

- `nova-light` → `nova`

- `nova-dark` → `nova-alt`

- `nova-colored` → `nova-accent`

Deprecations

- Responsive options deprecated from components like Table in favor of custom implementations.

- Label of RadioButton and Checkbox is deprecated; provide your own label instead.

- `MultiSelect.defaultLabel` is deprecated in favor of `placeholder`.

- SlideMenu is deprecated.

- Lightbox is deprecated in favor of Galleria.

- Spinner is deprecated in favor of InputNumber.

- `p-header` and `p-footer` are deprecated in favor of `` or ``.

PrimeNG Migration: Version 11 → Version 12

Overview

PrimeNG 12 is primarily a compatibility release for Angular 12, with one major incremental change in version 12.1 affecting the DataTable component's scrolling and column features. This guide covers migrating from PrimeNG v11.x to v12.x, detailing both v12.0 and v12.1 changes.

1. PrimeNG v12.0.0 (Compatibility Release)

- Angular 12 support: updated peer dependencies for @angular/core, @angular/cli, etc. - No breaking changes in core components compared to v11. All module imports, CSS classes, and APIs remain the same. - New minor components and bug fixes were introduced, but none require migration work.

2. PrimeNG v12.1.0 (DataTable Scrolling Overhaul)

Version 12.1 introduces a rewritten DataTable (p-table) scrolling mechanism using CSS Sticky. This enhances performance and simplifies the DOM, but requires changes to templates and styles.

2.1 Breaking Changes in p-table (DataTable)

- **Removal of support:** Templates using inside will no longer function. Remove elements from your table templates. - **Frozen Columns:** The frozenColumns input property is removed. Use the pFrozenColumn directive on tags to mark columns as frozen. - **Template Renaming:** The frozenRows template slot is renamed to frozenBody. Update declarations accordingly. - **scrollDirection:** A new scrollDirection input replaces older boolean flags for vertical or horizontal scrolling. Use values "vertical", "horizontal", or "both".

2.2 Example Template Migration

Before (v11):
<p-table [value]="data" [scrollable]="true" [scrollHeight]="'400px'"> <colgroup> <col style="width:200px" /> <col style="width:200px" /> </colgroup> <p-column field="name" header="Name" frozenColumns></p-column> <p-column field="age" header="Age"></p-column> </p-table>

After (v12.1):
<p-table [value]="data" scrollable="true" scrollHeight="400px" scrollDirection="both"> <p-column pFrozenColumn field="name" header="Name" style="width:200px"></p-column> <p-column field="age" header="Age" style="width:200px"></p-column> </p-table>

3. Removal of FullCalendar Component

The p-fullCalendar component included in PrimeNG v9–v11 is deprecated and removed in v12.1.0. To maintain calendar functionality, integrate the official FullCalendar Angular wrapper: - Install: npm install @fullcalendar/angular @fullcalendar/daygrid - Import FullCalendarModule from '@fullcalendar/angular'. - Update templates to use instead of .

4. Styling and Class Names

There are no new class prefix changes in v12. All p- classes from v10 remain. However, verify any custom styles targeting table scroll wrappers, as the internal DOM structure changed.

5. Verification & QA

- Run regression on all tables with scrolling and frozen columns. - Update unit/E2E tests for changed template slots and inputs. - Visual regression to catch style differences in table headers, footers, and scroll bars. - Cross-browser testing for sticky positioning support (modern browsers only).

6. Step-by-Step Migration Plan for v12

1. Update PrimeNG package: npm install primeng@^12.1.0 primeicons@^4.1.0. 2. Remove usage from templates. 3. Replace frozenColumns inputs with pFrozenColumn directives. 4. Rename frozenRows templates to frozenBody. 5. Update scroll inputs: use scrollDirection instead of boolean props. 6. Migrate FullCalendar usages to @fullcalendar/angular. 7. Test and verify all table variants. 8. QA and deploy updated UI.

PrimeNG Migration: Version 12 → Version 13

Overview

PrimeNG 13 introduces Ivy partial compilation and removes support for ViewEngine and Internet Explorer 11. This release is largely backward-compatible for component APIs and class prefixes, but changes development requirements and default theming.

1. Ivy-Only Compilation

- PrimeNG 13 ships with partial Ivy compilation output. ViewEngine is no longer supported, aligning with Angular 13's removal of ngcc. Ensure your application is running under Ivy (Angular 12+ default).

2. Dropped IE11 Support

- Official support for Internet Explorer 11 is removed. CSS and polyfills for IE11 no longer included in PrimeNG v13 themes. Remove any IE-specific hacks or polyfills from your project.

3. Default Theme Change to Lara

- PrimeNG v13 adopts the new 'Lara' theme as the default. If you were using an older theme (Nova, Luna, Saga), you must explicitly import your preferred theme CSS.

Example Theme Import in angular.json: ```json "styles": ["node_modules/primeng/resources/themes/lara-light-indigo/theme.css", "node_modules/primeng/resources/primeng.min.css", "node_modules/primeicons/primeicons.css", "src/styles.scss"] ```

4. Component and Class API Stability

There are no new breaking changes to component templates or CSS class prefixes. All p- classes remain consistent with v12. Verify that any custom CSS still targets the correct p- selectors.

5. Removal of Deprecated APIs

- Any leftover deprecated APIs from PrimeNG <=11 have been removed. For example, ensure you are not using `PrimeNGConfig.ripple` if removed or other removed methods. Refer to the changelog for a list of removed methods and inputs.

6. Dependency Updates and Requirements

- Angular 13 required: Update peer dependencies: @angular/core@^13.0.0, @angular/cli@^13.0.0. - PrimeNG 13 requires PrimeIcons >=4.1.0 and may require PrimeFlex >=3.0.0 for updated utilities.

7. Verification & QA Steps

- Ensure application builds under Ivy without ngcc. - Remove IE11 polyfills and test in supported browsers only (Chrome, Firefox, Edge, Safari). - Review visual styling: confirm theme CSS is correctly applied (Lara or custom

choice). - Run unit and E2E tests to catch any missing providers or polyfill errors.

8. Migration Plan for v13

1. Update package.json: - primeng to ^13.0.0 - @angular/core and @angular/cli to ^13.0.0 - primeicons to ^4.1.0 - primeflex to ^3.0.0 2. Remove IE11 polyfills from polyfills.ts. 3. Ensure tsconfig.json has enableIvy: true (default in Angular 13). 4. Update angular.json to import desired theme CSS (Lara or custom). 5. Build and address any compiler errors related to missing ngcc steps. 6. Run tests and QA. 7. Deploy.

PrimeNG Migration: Version 13 → Version 14

Overview

PrimeNG 14 is an incremental release focused on compatibility with Angular 14 and performance improvements. It includes updates to virtualization components and minor refinements, but no breaking changes to CSS class prefixes or core APIs.

1. Compatibility and Dependency Updates

- Angular 14 support: Ensure peer dependencies for @angular/core, @angular/cli are ^14.x. - PrimeIcons should remain at >=4.1.0. PrimeFlex version should align with PrimeNG documentation (>=3.x).

2. Virtual Scrolling Enhancements

PrimeNG 14 introduces an updated VirtualScroller for faster rendering of large lists in components like DataTable, Dropdown, and MultiSelect. The API remains the same but internal performance is improved. No template changes are required. If you use [virtualScroll] or [virtualRowHeight], simply upgrade the package and rebuild to benefit from optimizations.

3. New and Enhanced Components

- Added minor features to existing components (e.g., new event hooks in Carousel and Galleria). - Introduced new components such as the TreeSelect for combined tree and dropdown selection. - Use TreeSelectModule and <p-treeSelect> tag.

4. No CSS or Structural API Breaks

There are no changes to CSS class prefixes (p- remains) or DOM structure for existing components. Verify that any custom CSS still targets the correct selectors, but expect no regressions.

5. Migration Steps for v14

1. Update package.json: primeng@^14.0.0. 2. Run npm install to update dependencies. 3. Rebuild the application and fix any peer dependency warnings. 4. Run unit and E2E tests to confirm behavior. 5. Optionally test new TreeSelect component and virtual scroll performance.

6. Verification & QA

- Focus on large data components (DataTable, Dropdown) to confirm improved scroll performance. - Test TreeSelect usage in forms for both single and multiple selection modes. - Run cross-browser tests to ensure virtualization works across supported browsers.

PrimeNG Migration: Version 15 → Version 20

Section 1: Migration v15 → v16

PrimeNG v16 focuses on Angular v16 compatibility and enhancements: - **Dependency Updates:** Update to `primeng@^16.0.0`, `primeicons@^5.1.0`, `primeflex@^3.3.0`. - **Theming:** Introduced dynamic theme loading API for CSS variables; use `PrimeNGConfig.set('theme', 'lara')` dynamically. - **Accessibility Improvements:** Enhanced ARIA roles on key components like Button, Table, and Dialog. - **Removal of Deprecated Classes:** Any legacy CSS variables or classes from v10–v14 removed; ensure SCSS overrides use new variable names. - **Migration Steps:** 1. Update `package.json` and run `npm install`. 2. Adjust theming initialization in `main.ts` or `app.module.ts`. 3. Rebuild and run tests. 4. Verify ARIA attributes and perform accessibility audit.

Section 2: Migration v16 → v17

PrimeNG v17 brings performance optimizations and new component variants: - **Performance:** VirtualScroller and Table optimizations reduce bundle size by code-splitting component modules. - **New Components:** Introduced `MegaMenu` with improved keyboard navigation. - **Breakpoints Update:** PrimeFlex updated grid breakpoints from `p-col-` classes to support CSS container queries optionally. - **Migration Steps:** 1. Update to `primeng@^17.0.0` in `package.json`. 2. Review import paths for Lazy-loaded component modules. 3. Update grid CSS classes if using container query mode. 4. Run build and tests; validate menu keyboard flows.

Section 3: Migration v17 → v18

PrimeNG v18 aligns with Angular v18 features: - **Signals Integration:** Components emit signals APIs for reactive rendering; opt into signals with `PrimeNGConfig.enableSignals()`. - **Zoneless Mode Support:** Dialog and Overlay components support zone-less operation. - **Deprecations:** Removed support for legacy dialog templates (`` / ``) entirely. - **Migration Steps:** 1. Bump to `primeng@^18.0.0`. 2. Enable signals in `app.module.ts`. 3. Replace any deprecated template tags with ``. 4. Execute application and validate change detection boundaries.

Section 4: Migration v18 → v19

PrimeNG v19 introduces SSR hydration improvements and standalone component support: - **SSR Hydration:** Templates support hydration-friendly markup for faster server-side rendering. - **Standalone Components:** UI components available as standalone exports for direct use without NgModule. - **Theme Builder:** CLI tool to generate custom themes via JSON configuration. - **Migration Steps:** 1. Install `primeng@^19.0.0` and `primeicons@^5.2.0`. 2. Update overlays to container-friendly hydration wrappers. 3. Migrate modules to standalone imports (`import {Button} from 'primeng/button'`). 4. Generate and include custom theme via `primeng theme` CLI.

Section 5: Migration v19 → v20

PrimeNG v20 stabilizes the builder and introduces Vue and React wrappers: - **Builder Enhancements:** PrimeNG CLI Builder supports incremental rebuilds and filesystem caching. - **Vue/React Support:** Experimental wrappers for Vue 3 and React 18 released; see `@primeng/react` and `@primeng/vue` packages. - **Deprecations:** Removed legacy PrimeFlex utility classes in favor of CSS utility modules (`@primeng/utils-css`). - **Migration Steps:** 1. Upgrade to `primeng@^20.0.0`, `primeicons@^5.3.0`. 2. Replace PrimeFlex references with `@primeng/utils-css` imports. 3. Evaluate Vue/React wrapper usage if

relevant to polyglot architectures. 4. Run full regression and cross-framework integration tests.

Angular Upgrade Guide: v15 → v18

Section 1: Angular v15 Baseline

Ensure your project is running Angular v15 before starting the upgrade path. Baseline requirements: - TypeScript 4.8+ - Node.js 14 or 16 - Angular CLI v15 - ngcc migration completed (Ivy enabled) Familiarize with new v15 features such as metaprogramming APIs and turbo-enabled build pipelines.

Section 2: Upgrade v15 → v16

Angular v16 introduces removal of ngcc, partial compilation support, and experimental signals. Prerequisites: - Ensure TypeScript is upgraded to 4.9 or above - Update Node.js to 16 or 18 (recommended) Upgrade Steps: 1. Run `ng update @angular/core@16 @angular/cli@16` 2. Confirm removal of ngcc in postinstall scripts 3. Test Ivy compilation without ngcc: `ng build` 4. Opt-in to signals preview by importing `@angular/core/rxjs-interop` 5. Update tsconfig to include "defineSignal" options if using signals 6. Execute unit and E2E tests 7. Address any breaking changes (e.g., library legacy ViewEngine removal)

Section 3: Upgrade v16 → v17

Angular v17 stabilizes control flow directives and esbuild-based builds. Prerequisites: - TypeScript 5.2+ - Node.js 18+ Upgrade Steps: 1. Run `ng update @angular/core@17 @angular/cli@17` 2. Address Type-only directive migration warnings for *ngIf and *ngFor 3. Update builders to esbuild by default, remove custom webpack configs if unused 4. Refactor any deprecated lifecycle hooks and remove renderComponent polyfills 5. Re-run tests and fix any template type errors (strict mode) 6. Verify production build size and performance improvements

Section 4: Upgrade v17 → v18

Angular v18 makes signals stable and previews zoneless mode. Prerequisites: - TypeScript 5.4+ - Node.js 18 or 20 Upgrade Steps: 1. Run `ng update @angular/core@18 @angular/cli@18` 2. Migrate to stable signals API if using preview signals 3. Experiment with zoneless mode via `import '@angular/platform-browser/zone-less';` 4. Update tsconfig.json for useNgDevMode flags and enable emitZone configurations 5. Remove any legacy Zone.js shims if opting into zoneless 6. Perform end-to-end testing for change detection boundaries 7. Validate SSR hydration improvements if applicable

Summary and Next Steps

After completing to v18, continue with sequential updates to v19 and v20. Perform incremental testing after each version upgrade, maintain version control branches, and leverage ng update schematics for safe migrations.

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.