

# Astro + shadcn/ui Integration Learnings

---

## Key Findings & Best Practices

### ✗ What DIDN'T Work Initially

#### 1. Astro CLI Integration Issues

```
# These commands installed packages but didn't update astro.config.mjs
npm run astro add react
npm run astro add tailwind
```

- **Problem:** Dependencies were installed but config file remained empty
- **Solution:** Use `--yes` flag and verify config was updated

#### 2. shadcn/ui Detection Problems

```
npx shadcn@latest init
# Error: No Tailwind CSS configuration found
```

- **Problem:** shadcn couldn't detect Tailwind even when installed
- **Root Cause:** Missing proper Tailwind config file structure

### ✅ What WORKED - The Correct Flow

#### 1. Proper Astro Integration Setup

```
# Use --yes flag for better automation
npx astro add react --yes
npx astro add tailwind --yes
```

**Critical:** Verify `astro.config.mjs` gets updated with:

```
import react from '@astrojs/react';
import tailwindcss from '@tailwindcss/vite';

export default defineConfig({
  integrations: [react()],
  vite: {
    plugins: [tailwindcss()]
  }
});
```

## 2. Astro Uses Tailwind v4 + Vite Plugin Approach

- **Not** the traditional `@astrojs/tailwind` integration
- Uses `@tailwindcss/vite` plugin directly
- CSS import: `@import "tailwindcss";` (not separate base/components/utilities)

## 3. Path Aliases Required for shadcn/ui




```
// tsconfig.json
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": [".src/*"]
    }
  }
}
```

**Note:** This is a manual configuration step - there is NO CLI tool for adding TypeScript path aliases. This is the correct prescribed approach per official documentation.

## 4. shadcn/ui Initialization

```
npx shadcn@latest init
```

### Detects:

-  Astro framework
-  Tailwind v4 configuration
-  Path aliases

### Creates:

- `components.json` configuration
- `src/lib/utils.ts` utility functions
- Enhanced `src/styles/global.css` with design tokens

## The Ideal Development Approach

### Keep Components as .astro Files

Instead of creating separate React components, use shadcn components directly in `.astro` files:

```
---
import { Button } from "@components/ui/button";
```

```
import { Card } from "@components/ui/card";
---

<section class="container mx-auto">
  <Button size="lg" client:load>Get Started</Button>
  <Card client:load>
    <!-- Content -->
  </Card>
</section>
```

### Use client: directives Strategically

- **client:load** - For interactive components
- No directive needed for static shadcn styling

### Manual Steps That Are Actually Correct

After investigation, these manual configurations have **NO CLI alternatives** and are the prescribed approach:

1. **TypeScript Path Aliases** - Must be manually added to **tsconfig.json**
2. **Global CSS Import** - Must be manually imported in **Layout.astro**

These are not workarounds but the official way to configure these features in Astro.

### Technical Architecture

#### Tailwind v4 Integration

- Uses new **@theme inline** syntax
- OKLCH color space for better color handling
- Design tokens through CSS custom properties
- Automatic dark mode support

#### Component Architecture

```
src/
├── components/
│   ├── ui/           # shadcn/ui components (React)
│   ├── Hero.astro    # Page components (Astro + shadcn)
│   └── Features.astro
├── lib/
│   └── utils.ts       # shadcn utilities (cn function)
└── styles/
    └── global.css     # Tailwind + design tokens
```

### Common Pitfalls to Avoid

1. **Don't mix CLI approaches** - Use Astro CLI for integrations, shadcn CLI for components
2. **Don't manually configure when CLIs work** - Let tools handle the config
3. **Don't create unnecessary React components** - Use .astro files with shadcn imports
4. **Don't forget client: directives** - Interactive components need hydration

### Perfect Setup Checklist

- ☐ `npx astro add react --yes`
- ☐ `npx astro add tailwind --yes`
- ☐ Verify `astro.config.mjs` was updated
- ☐ **Manually** add path aliases to `tsconfig.json` (no CLI available)
- ☐ **Manually** import global CSS in `Layout.astro` (no CLI available)
- ☐ `npx shadcn@latest init`
- ☐ `npx shadcn@latest add [components]`
- ☐ Import components in `.astro` files
- ☐ Add `client:load` to interactive components

### Benefits of This Approach

- **Simple**: Minimal setup, maximum functionality
- **Fast**: Astro's zero-JS default + shadcn's performance
- **Flexible**: Mix static and interactive as needed
- **Modern**: Latest Tailwind v4 + React patterns
- **Maintainable**: Clear separation of concerns

### Key Insights

1. **Not Everything Has a CLI** - Some configurations like TypeScript path aliases and CSS imports are meant to be done manually. This is by design, not a limitation.
2. **Follow the Prescribed Flow** - Use Astro CLI for integrations (`astro add`), shadcn CLI for components (`shadcn add`), and manual configuration where documented.
3. **The Power of Simplicity** - The combination of Astro's file-based routing + islands architecture with shadcn/ui's component system creates an incredibly powerful development experience. You get the performance benefits of static generation with the developer experience of modern component libraries.