CASCADIAJS 2026

Shared Components Beyond the Design System

Where we're headed

01

What's a design system?

02

Why go beyond?

03

Who are you building for?

04

Where are you building it?

05

Building it thoughtfully

06

Keeping it maintainable

07

Fostering an ecosystem

What's a Design System?

Central Platform
Usually owned by a central team, shipped as a unified platform

Atomic components
Often basic components like buttons, dropdowns, modals

Primitives & tokens
Includes primitives like the color palette, spacing system, and typography

But a design system can’t do everything

Beyond the design system

Navigation &
higher-level UX

Team-specific
components

Design system
candidates

Who are you building for?

Anchor consumer
The team for whom you're building right now — might even be your team

Second consumer
A real or imaginary second team — to ensure you keep your component generic

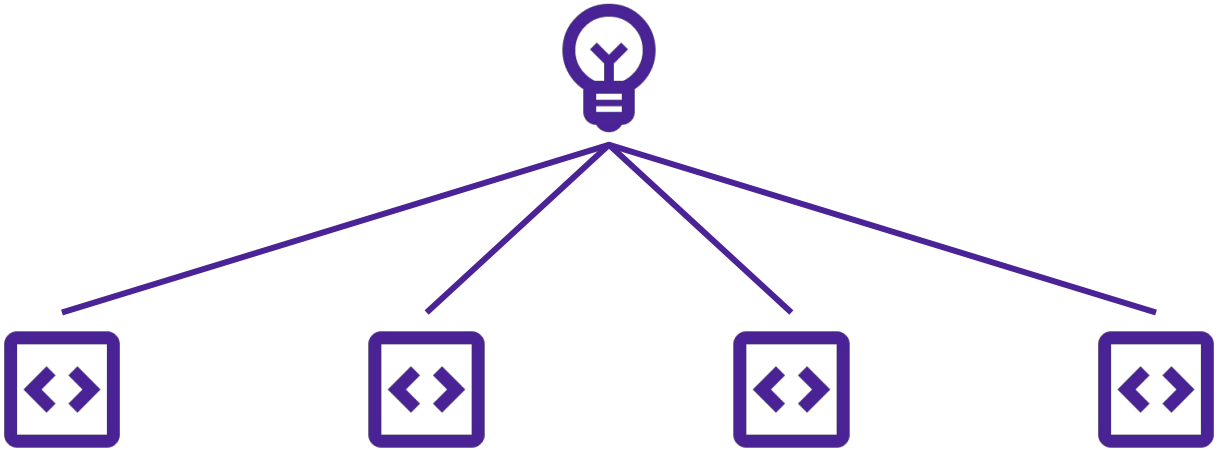
Where are you building it?

Keep it separate
Don’t put it in your app repo — keep it separate so it can evolve independently

Publish artifacts
Release to npm/artifactory/etc so other teams can use your component freely

Building it **thoughtfully**

Build up from the design system



Make your APIs consistent

BASIC TEXT INPUT

```
value: string;
onChange: (newValue: string) => void;
size: “small” | “default” | “large”;
placeholder: string;
maxLength: number;
...
```

SNAZZY SEARCH BOX

```
datasourceUrl: string;
maxOptions: number;
onSubmit: (value: string) => void;
value: string;
onChange: (newValue: string) => void;
size: “small” | “default” | “large”;
placeholder: string;
maxLength: number;
...
```

Design with **Layered Extensibility**

Make the right way the easy way
Optimize the API for the 80% case, providing sensible defaults where possible

Offer layers of deeper customization
Don’t make consumers eject into hard mode just to get a slightly custom experience

Design with Layered Extensibility

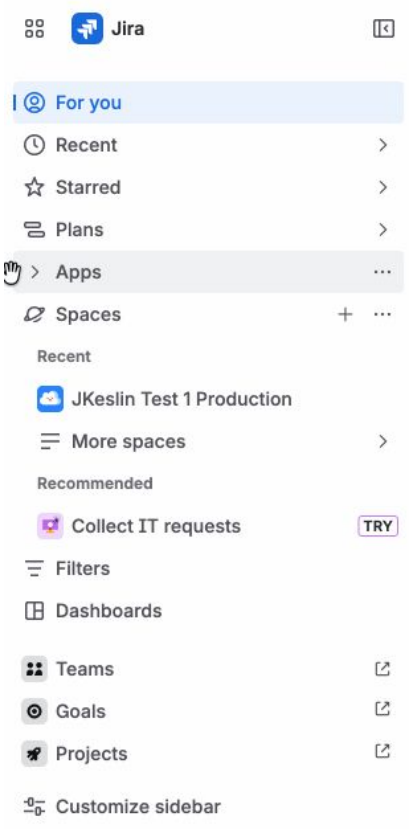
80% CASE: TEXT


```
<Button
  label="My Awesome Button"
/>
```

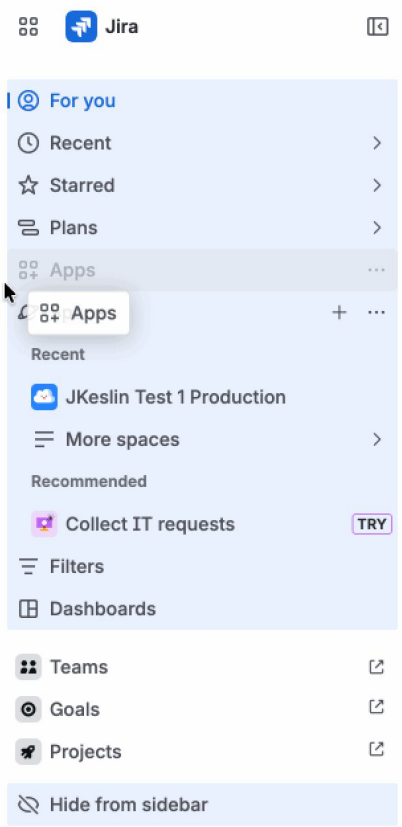
15% CASE: REACT NODE


```
<Button
  label="My Awesome Button">
  <HStack>
    <Icon type="star" />
    My Awesome Button
  </HStack>
</Button>
```

Example: Jira's navigation

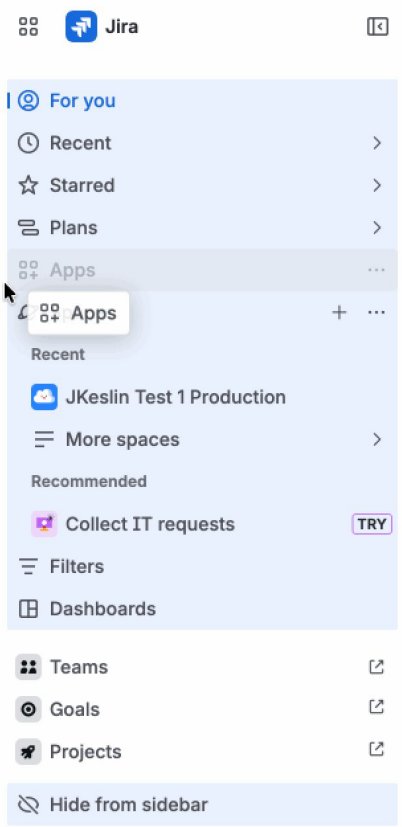


Example: Jira's navigation




```
// Common case - fully declarative
{
  type: “link”,
  label: “For you”,
  icon: <YouIcon />,
  href: “/for-you”
}
```

Example: Jira's navigation




```
// Custom items - functional
{
  type: “custom”,
  label: “Apps”,
  icon: <AppsIcon />,
  component: AppsNavItem
}
```

```
// AppsNavItem props
{
  draggableRef: ReactRef,
  dropIndicator: ReactNode,
  isDragging: boolean,
  ...
}
```

Optimize for the higher layers

Consider common use-cases
Make it easy to do typical things like setting text labels, even when using intl libraries

Don’t forget complex but frequent things
Integrating analytics or spotlight tour elements shouldn’t require ejecting to hard-mode

Don’t build a Homer Car

It’s okay to
be opinionated

Consider including batteries

Handle data fetching, if you can
Use mechanisms like Relay's fragments, for example, to encapsulate data requests

Reduce boilerplate as much as possible
Using your component shouldn't require writing a novel — keep it simple

Keep an escape hatch, though
Despite your best efforts, sometimes a consumer needs more control

Make it **maintainable**

Write thorough tests
Keep quality high — write plenty of unit and integration tests so your component is rock solid

Document liberally
You won’t always be there for support — document everything

Keep a changelog
Updates happen — keep your consumers appraised of what changed when

Foster an **ecosystem**

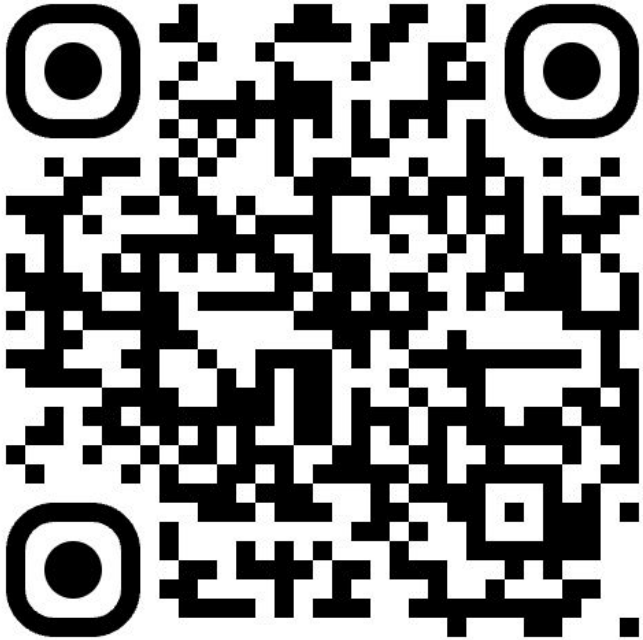
Shared components are super valuable
They de-duplicate effort, become a force multiplier, and improve consistency across apps

Governance is essential
Establish patterns and practices for consistent, high-quality results

Discoverability makes a big difference
Investing in a common doc site / marketplace for shared components pays dividends

Now, go build something **awesome!**

Thank you!



jonathankeslin.com/cascadiajs26