Component libraries are often used in component-based UI systems like React and Vue. They are typically versioned repositories of components.

IBM's Carbon Design System and Palantir's Blueprint are both good examples.

Why component libraries

There are several rationales for creating component libraries.

- **Create unified design**. In large web properties and web apps, the look and feel can diverge across different sections maintained by different teams. Component libraries are often used to implement a design system.
- **Avoid reinventing the wheel**. Component libraries include common elements like carousels or dropdowns to avoid the need for individual teams to reimplement these components.

Tooling & team setups

Component libraries are typically maintained by one individual or a design team that acts as a curator; when a website team or feature team needs a component, it is typically available ready for them to use, so they can move faster.

Component libraries are typically stored in a separate repository. Individual apps or websites then specify in their dependencies (in `package.json`) which version of each component they are using.

One drawback of using component libraries is the additional complexity of cross-repository dependencies.

For example, if a feature developer need to change a library component, that developer's workflow typically involves two pull requests; one to the component repository repo to make the changes, and one to the website repository to bump the component version.

Different versioning approaches

There are two different approaches for versioning component libraries.

The first is to version on globally across the component library. At any given commit, the library has one version number (e.g. `30.3.1`). Any commit updating a component will then bump the version number accordingly. Both Carbon Design System and Blueprint take this approach.

The second approach is to version each component in the component library. This was used, for example, by Walmart.com -- they built their component library as React components, and created every component as a separate, versioned npm package.

This approach allows more granularity -- *what if you want an older version of one component, but a newer version of another one?* -- but requires additional tooling to make developer workflows pleasant.