ANGULAR COMPONENT-BASED DESIGN STRATEGY

VIVEKANAND P V

PRELIMINARIES

- *Components define areas of responsibility in the user interface, or UI, that let you reuse sets of UI functionality* (from documentation)
- Component is a patch on the view (browser) that has structure (template), style (css), behavior (class), and testability (spec)
- Generally, a component is a set of html, css, ts, and spec.ts files all packed in its own folder

CORE CONCEPTS

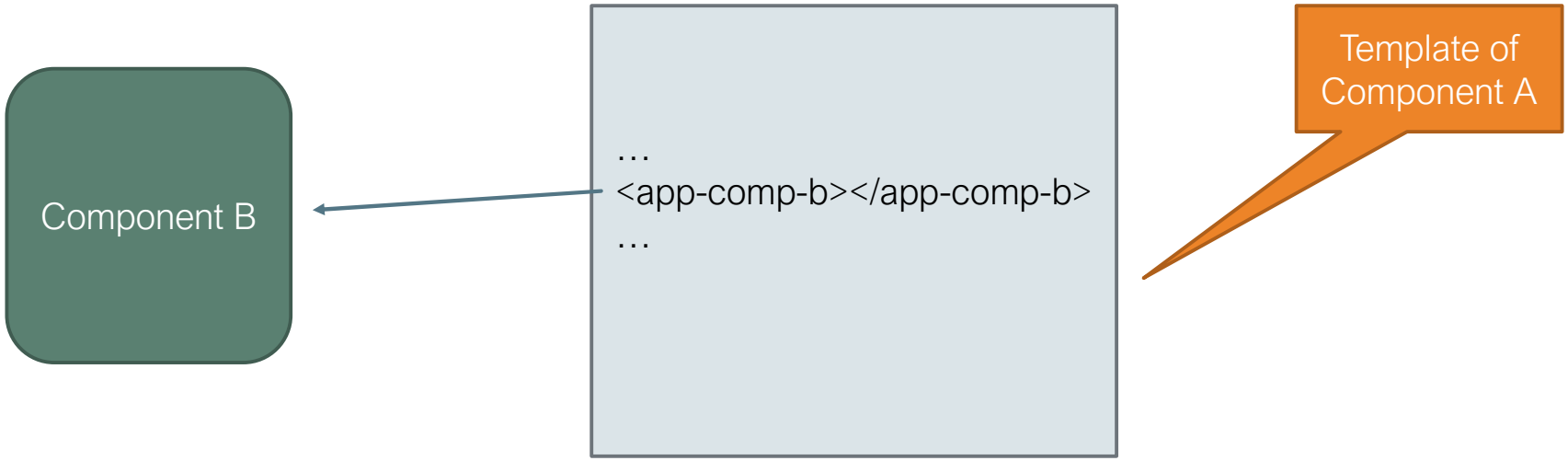
- Components are declared in a module (root module or a feature module) and they are private to the declaring module unless they are exported
- A component and its template (html file) are linked via templateUrl property
- A component and its custom tag in html (e.g., <app-item>) are linked via selector property

ROOT COMPONENT

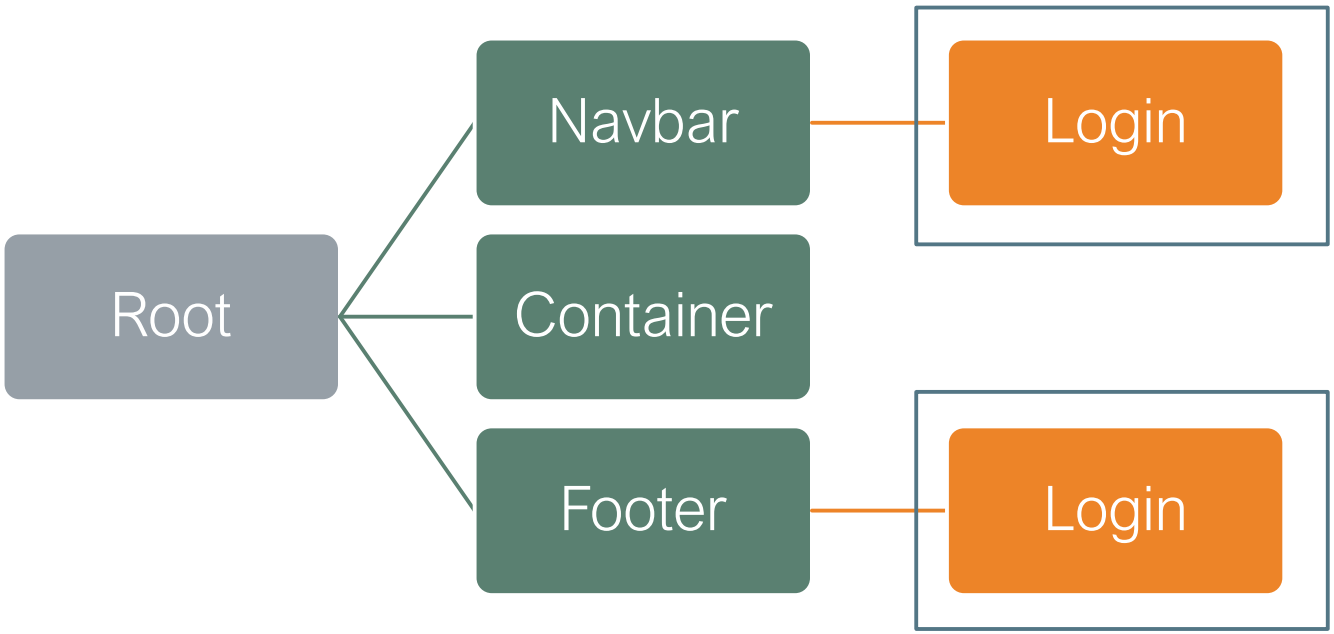
- AppComponent is the one created by the CLI, which is called as the root component. This has the selector <app-root>
- Root component is the one that is bootstrapped by the root module (i.e., AppModule)
- Root component sets up the component hierarchy for the application

COMPONENT COMPOSITION

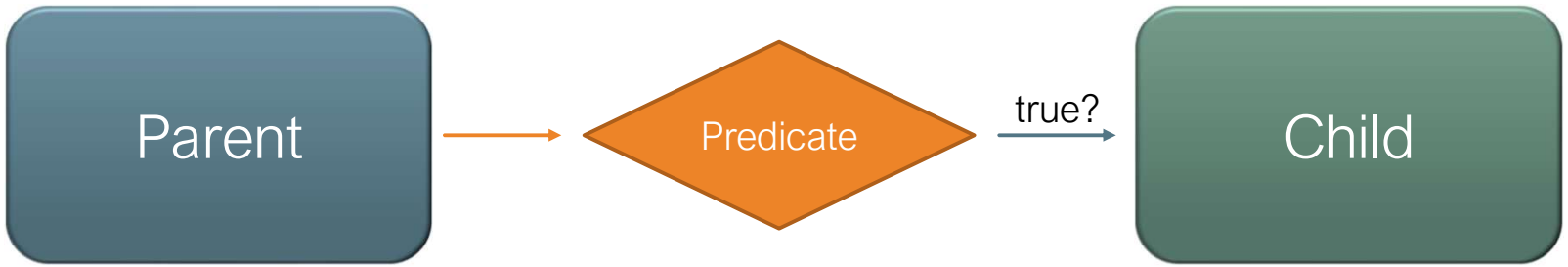
If component A uses component B in its template, we can say A is the parent while B is the child



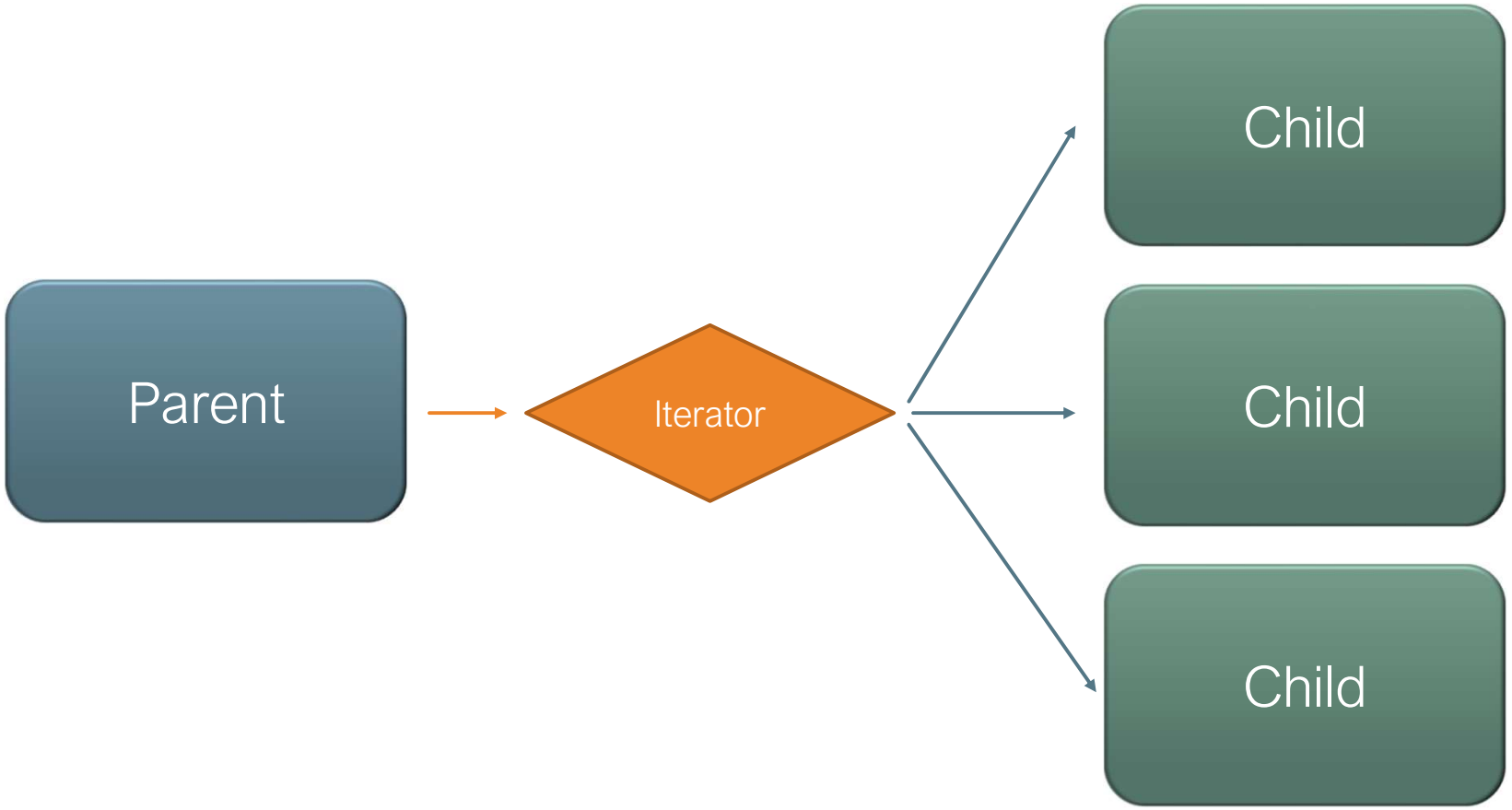
COMPOSITION STRATEGIES: REUSABILITY



COMPOSITION STRATEGIES: CONDITIONALITY



COMPOSITION STRATEGIES: MULTIPLICITY



PLEASE NOTE

- Do not change the component's selector to a short, general word
- Beware of circular reference in component (e.g., A > B > C > A). Angular runtime detects it and errors after reaching a stack limit.
- Design components with reusability in mind. Follow good software practices such as DRY and SOLID.
- Blend the designer's and engineer's perspectives for better component strategies
- Refactor whenever necessary