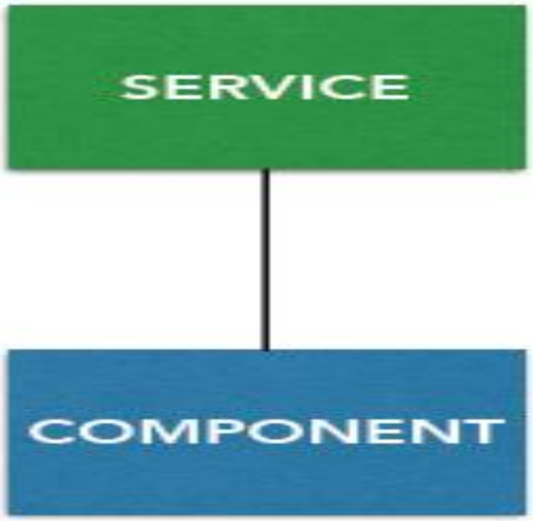
Redux

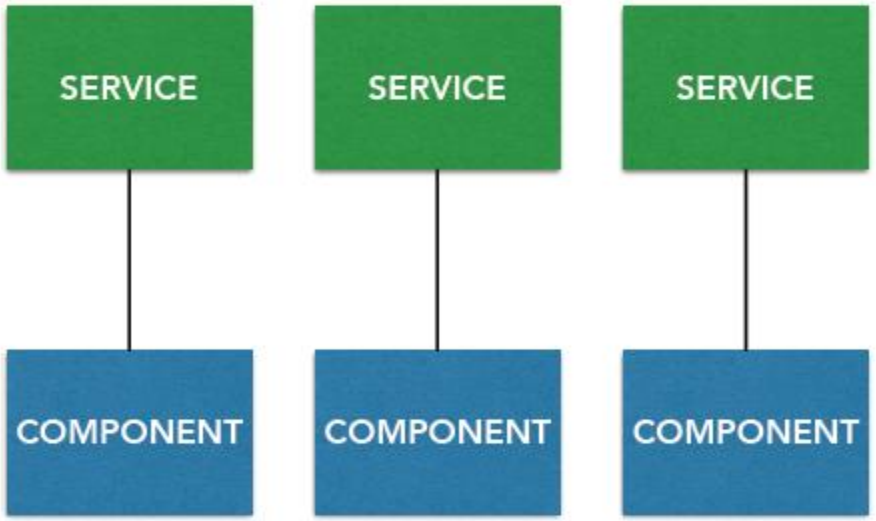
Redux

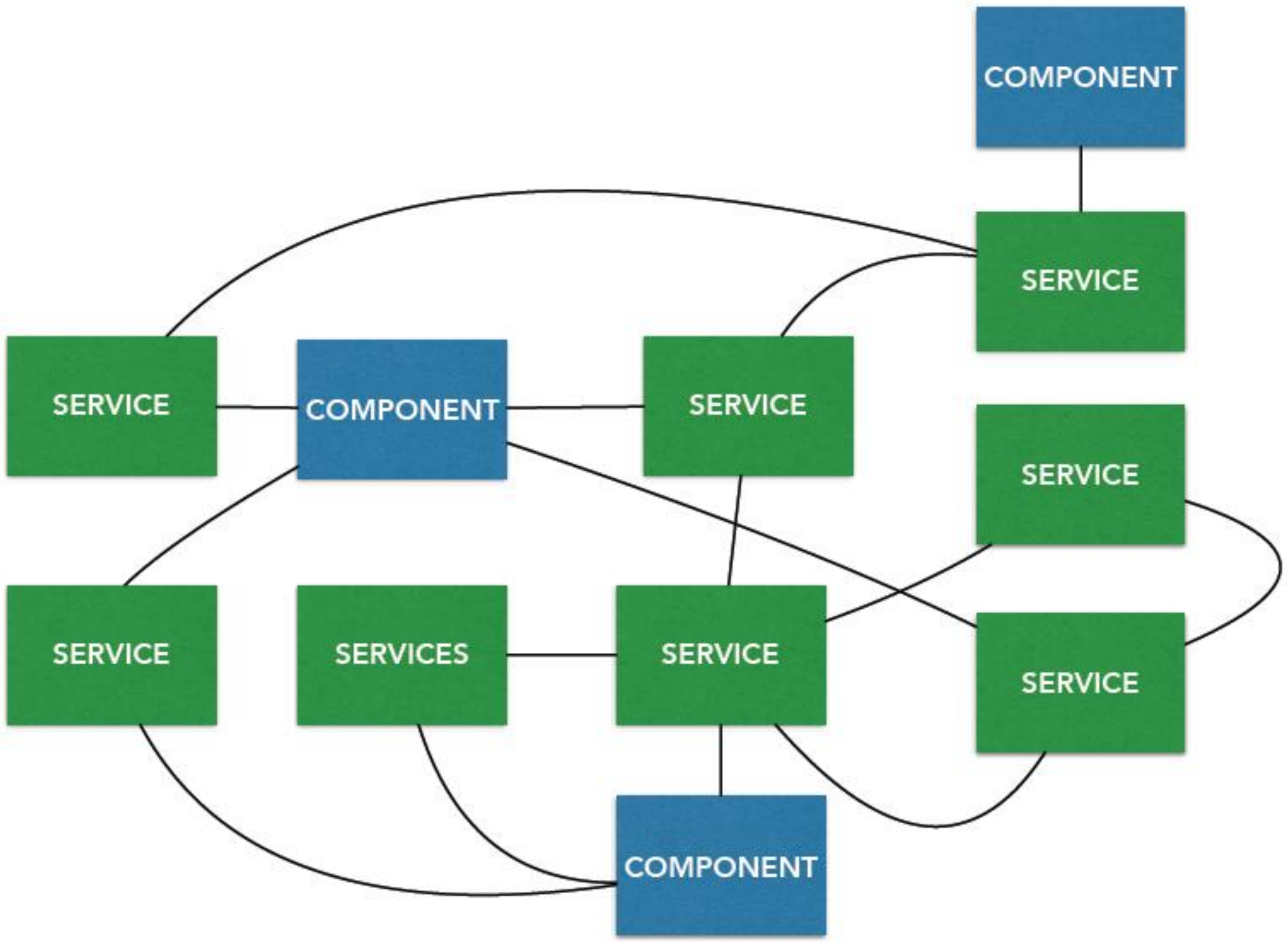
- Redux is a predictable state container for JavaScript application. It helps you write application that behave consistently, run in different environments (client, server and native).
- Simply put, Redux is a state management tool.
- With Redux, the state of your application is kept is a store and each component can access any state that it needs from this store.

Continue...

- In Angular if we can share the data between component using @Input/@Ouput.
- In React JS we can share the state(data) between two component using props.
- When the application is very big then it is very difficult to share the data between more than one components.
- Redux is a pattern/library from the React world. The purpose of Redux is to make application data more predictable by creating a one-way data flow.


SERVICE
COMPONENT


SERVICE
SERVICE
SERVICE
COMPONENT
COMPONENT
COMPONENT


COMPONENT
SERVICE
SERVICE
COMPONENT
SERVICE
SERVICE
SERVICE
SERVICES
SERVICE
SERVICE
COMPONENT

CHANGING SOMETHING

BREAKS

SOMETHING SOMEWHERE

REDUX

Predictable state container for JavaScript apps.

SINGLE SOURCE OF TRUTH

"The state of your whole application is stored in an object tree within a single store"

Building parts in Redux

- 1. Store
- 2. Reducers
- 3. State
- 4. Actions

Store in Redux :

- To put simply, store is the "database" of our application. It comprises of different states defined in our application. The state, thus, is immutable and only altered by actions.
- The store hold the application state. There is only one store in any Redux application. your store can be thought of as a client side "single source of truth".
- You can access the state stored, update the state and register or unregister listeners via helper methods.

Reducer in Redux

- If the store is the database of the application, the reducers are the tables.
- Reducer are pure function that take the current state of an application perform an action and return a new state.
- The state are stored as object and they specify how the state of an application changes in response to an action sent to the store.
- Reducers take the previous state of the app and return a new state based on the action passed to it.

State in Redux

- State is a single immutable data structure. States are what makes up the store.
- As stated before, the reducers are like tables, and thus state are fields in the table.