State and storage Management

**Analysis by**: Pranjal Karankar

**Target Audience:** Front End Development Team

1. Pranjal Karankar
2. Chiraag Subramanian

**Optional Audience:**

Team 6: All members

**Why is it required?**

Large applications like our “Show Me” which are heavily focused on the client-side implementation of logic, often grow in complexity extremely fast.

It becomes very hard to manage state of the application or any member variables in complex application where a lot of components need to talk to each other, share information. If the components are related that is within parent-child relationship, it is somehow easy to share information. However, when components are completely unrelated, it becomes very hard to share information and manage state of mutually shared variables.

To address multiple pieces of state scattered across many components and the interactions between them, Vue offers Event bus mechanism and Vuex-our own Elm-inspired state management library.

**Available Options**:

1. Event Bus Mechanism
2. Vuex library

**Event bus**

After studying Event bus from following sources, I conclude:

The event bus is used more for communication between isolated components. So, in the event when two sibling components need to communicate with each other via emitting events and then listening for them. Events that don't necessarily mutate state or application data.

Sources studied for event bus implementation:

<https://alligator.io/vuejs/global-event-bus/>

<https://medium.com/@andrejsabrickis/https-medium-com-andrejsabrickis-create-simple-eventbus-to-communicate-between-vue-js-components-cdc11cd59860>

**Tutorials followed:**

Event Handling in Vue - Vue.js 2.0 Fundamentals: <https://www.youtube.com/watch?v=4PXXQzME5no>

# Vue JS 2 Tutorial #25 - The Event Bus: <https://www.youtube.com/watch?v=jzh4zQcfB0o>

**Vuex:**

Vuex = **state management pattern + library**

* a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion.

**Parts**:

* **state**- data source that drives our app;
* **view**- declarative mapping of the state;
* **actions**- which are the possible ways the state could change in reaction to user inputs from the view.

**When it can come into picture?**

**Problems**:

1. Multiple views may depend on the same piece of state.
2. Actions from different views may need to mutate the same piece of state.

**problem one**- passing props can be tedious for deeply nested components, and simply doesn't work for sibling components.

**problem two**- we try to address this problem by creating parent-child relationships and creating direct parent/child instance references or trying to mutate and synchronize multiple copies of the state via events. This quickly lead to unmaintainable code.

“why don't we extract the shared state out of the components, and manage it in a global singleton? With this, our component tree becomes a big "view", and any component can access the state or trigger actions, no matter where they are in the tree!”

**When to use Vue js**:

“If you've never built a large-scale SPA and jump right into Vuex, it may feel verbose and daunting. That's perfectly normal - if your app is simple, you will most likely be fine without Vuex. A simple event bus may be all you need. But if you are building a medium-to-large-scale SPA, chances are you have run into situations that make you think about how to better handle state outside of your Vue components, and Vuex will be the natural next step for you.”

(ref: https://vuex.vuejs.org/en/intro.html)

Followed two tutorial playlists on youtube:

<https://www.youtube.com/playlist?list=PL55RiY5tL51pT0DNJraU93FhMzhXxtDAo>

<https://www.youtube.com/playlist?list=PL4cUxeGkcC9i371QO_Rtkl26MwtiJ30P2>

**Conclusion**:

**Use case of Vuex in Our Project:**

As we grow our application, we are planning to create reusable components like session state manager, pagination components. It will be vital to share state of a member variable with all the components in the application. Event bus is good enough to achieve the task. However, it will soon become complex to mutate the data and pass that information to all components.

On the other hand, Vuex offer much better approach to state management. It performs data mutations in predictable fashion. It has a steeper learning curve as compared to Event Bus, and it is not easy to master. But it is REQUIRED. Hence, we decided to learn and start its implementation with baby steps.