Module 3 Day 14

Nested Components and Vuex

Nested Components

- Applications are composed of a tree of nested components
- Components should have a "single-purpose"
- To nest components
  - Create a new component
  - Define **props** to indicate what data the component needs to operate
  - Import the component into the parent component
  - Display the child on the parent, passing data into **props** using **attributes**

Sharing Data between Components

- props allows data to be "passed down"
  - From parent into child
  - It's reactive; if data changes on parent, it is reflected in the child
- However, props is one-way (parent to child)
- In a complex system
  - There are many components: parents, children and siblings
  - Data can be copied many times
  - It get exceedingly difficult to keep every component in sync when data changes

State Management with Vuex

- "Shared state" – All components have access to the "single source of truth"
- Isn’t this Global data? Isn’t Global data a maintenance nightmare?
  - Yes and yes! Especially when data is updated in multiple places.
  - It gets very difficult to determine who changed the data and why
- That’s why Vuex is referred to as a “pattern” as well as a tool
  - The difference between “**global data**” and “**managed state**”
  - All updates *go through* Vuex
  - These are called mutations

Not Unlike a DAO

- It’s an interface to get application data
- It’s an interface to update application data
- All reads and updates **should** go through mutations
  - Just like all reads/writes to the DB **should** go through DAOs
- It also allows components to not think about where the data comes from or goes.
  - Again, like a DAO
  - Next week, we will do API fetches from the Vuex mutations

Should All Data be in Vuex?

- Some data should remain private to the component
- Truly core application data
- Data that will be fetched from an API
- Data that represents a user's "current state"
  - Account information and role
  - A filter on views
    - For example, in a calendar app, the Date the user is currently viewing

Using Vuex

- Store/index.js
  - State: the shared data
  - Mutations: Any changes that anyone will make to state go here
- In a component
  - Reference data with
    - $store.state.mySharedVariable
    - this.$store.state.mySharedVariable
  - Change data (“commit a mutation”) with
    - this.$store.commit("UPDATE_FILTER", 0);