Tour of the App

In this first Watch Us Build lesson Gregg Pollack works alongside Damian Dulisz, a Vue Core team member, to build a Trello like clone. Damian is also known for his work on the Vue-multiselect library and Vuelidate libraries.

In this course, we will be building an application together step by step including:

- Setting up Vuex to read & write.
- Saving state to local storage
- Using the browser drag and drop API
- Refactor big components into smaller ones
- Creating reusable components
- Additional refactoring

We welcome you to code along with us, which you can do by cloning this repo and checking out the 'application-start' tag with `git checkout application-start`, or just downloading this startup code. We will provide you with all the code we are writing below each video, and if you get stuck the finishing code for each level is tagged appropriately.

I think you'll find Damian's approach to refactoring components extremely useful and implementing the browser's drag and drop API is fun to watch. In the next lesson we will get familiar with the code base, and start hooking up our data out of Vuex.

Building our Board

In this lesson we start by taking a tour of our starting code. Please feel free to download and follow along if you like. We'll be building out our initial Trello Board columns and tasks by pulling it out of Vuex. We'll also learn how to save and load our Board's state from our browser's localstorage.

Feel free to copy paste the code below if you get stuck, look at the differences in GitHub, or simply clone the repo and check out the lesson-2-complete tag.

/src/views/Board.vue

```
<template>
  <div class="board">
    <div class="flex flex-row items-start">
      <div
        class="column"
        v-for="(column, $columnIndex) of board.columns"
        :key="$columnIndex"
      >
        <div class="flex items-center mb-2 font-bold">
          {{ column.name }}
        </div>
        <div class="list-reset">
          <div
            class="task"
            v-for="(task, $taskIndex) of column.tasks"
            :key="$taskIndex"
          >
            <span class="w-full flex-no-shrink font-bold">
              {{ task.name }}
            </span>
            <p
              v-if="task.description"
              class="w-full flex-no-shrink mt-1 text-sm"
            >
              {{ task.description }}
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
import { mapState } from 'vuex'

export default {
  computed: mapState(['board'])
}
</script>
```

We'll also create a small `saveStatePlugin` so we can listen for mutations being made to our Vuex store, and store them in the local browser storage. This way if we refresh our browser, our board won't get reset.

/src/utils.js

```
...
export function saveStatePlugin (store) {
  store.subscribe(
    (mutation, state) => {
      localStorage.setItem(
        'board',
        JSON.stringify(state.board)
      )
    }
  )
}
```

We'll then need to use this plugin and restore the board from local storage if it already exists.

/src/store.js

```
import Vue from 'vue'
import Vuex from 'vuex'
import defaultBoard from './default-board'
import { saveStatePlugin } from './utils' // <-- Import saveStatePlugin

Vue.use(Vuex)

const board = JSON.parse(localStorage.getItem('board')) || defaultBoard

export default new Vuex.Store({
  plugins: [saveStatePlugin], // <-- Use
  state: {
    board
  },
```

In the next lesson we'll open up our tasks in a modal. See you then!