The Current Problem

NEXT: STATE MANAGEMENT

Before we understand what Vuex is and why it exist, we need to know what component composition is.

Components are like functions -- in fact, they are functions. Programming is solving problems by composing smaller functions. Components alike. The question becomes -- how do we pass in arguments and how do components interact with its environment. Just like parameters and return values in functions.

In Vue, we make use of `props` to pass data to components and component events to interact with the surrounding environment.

Table of Contents

An example of `props`:

```js
// ChildComponent
{
  // declare the props
  props: ['message'],
  created() {
    // Access props value from template
    console.log(message)
  },
  // bind props value to template
  template: '<div>{{ message }}</div>'
}
```

```js
// Parent Component
{
  template: `<child-component message="'hello'"></child-component>`,
  components: { ChildComponent }
}
```

The message prop is passed from the parent component down to the child component via a template.

Now let's see events example. Taking from what we have on the Vue site:

```js
var ButtonCounter = {
  template: '<button v-on:click="increment">{{ counter }}</button>',
  methods: {
    increment: function () {
      this.$emit('increment')
    }
  },
}
var ParentComponent = {
  el: '#counter-event-example',
  data: {
    total: 0
  },
  methods: {
    incrementTotal: function () {
      this.total += 1
    }
  },
  components: { ButtonCounter }
}
```

```html
<button-counter v-on:increment="incrementTotal"></button-counter>
```

When the button is clicked, the `ButtonComponent` emits an event. The parent component listens to this event and calls `incrementTotal` method.

So What is The Problem?

This is a good approach for moving data around. There is just this issue of components being nested as deep as 3+ levels and coordinating interaction becomes difficult.

When that's the case, you have to pass down data from component A to B, then B passes to C, and C to D, etc. Likewise, events, D emits to C, C to B, and so on until the component to handle the event is encountered.

Let's see a better way to tackle this situation.