



FÁBIO VEDOVELLI

Head of Front End Development at octimine Technologies
Munich, Germany

- . web developer
- . php e javascript
- . laravel e vue.js

Twitter: [@vedovelli](https://twitter.com/vedovelli)

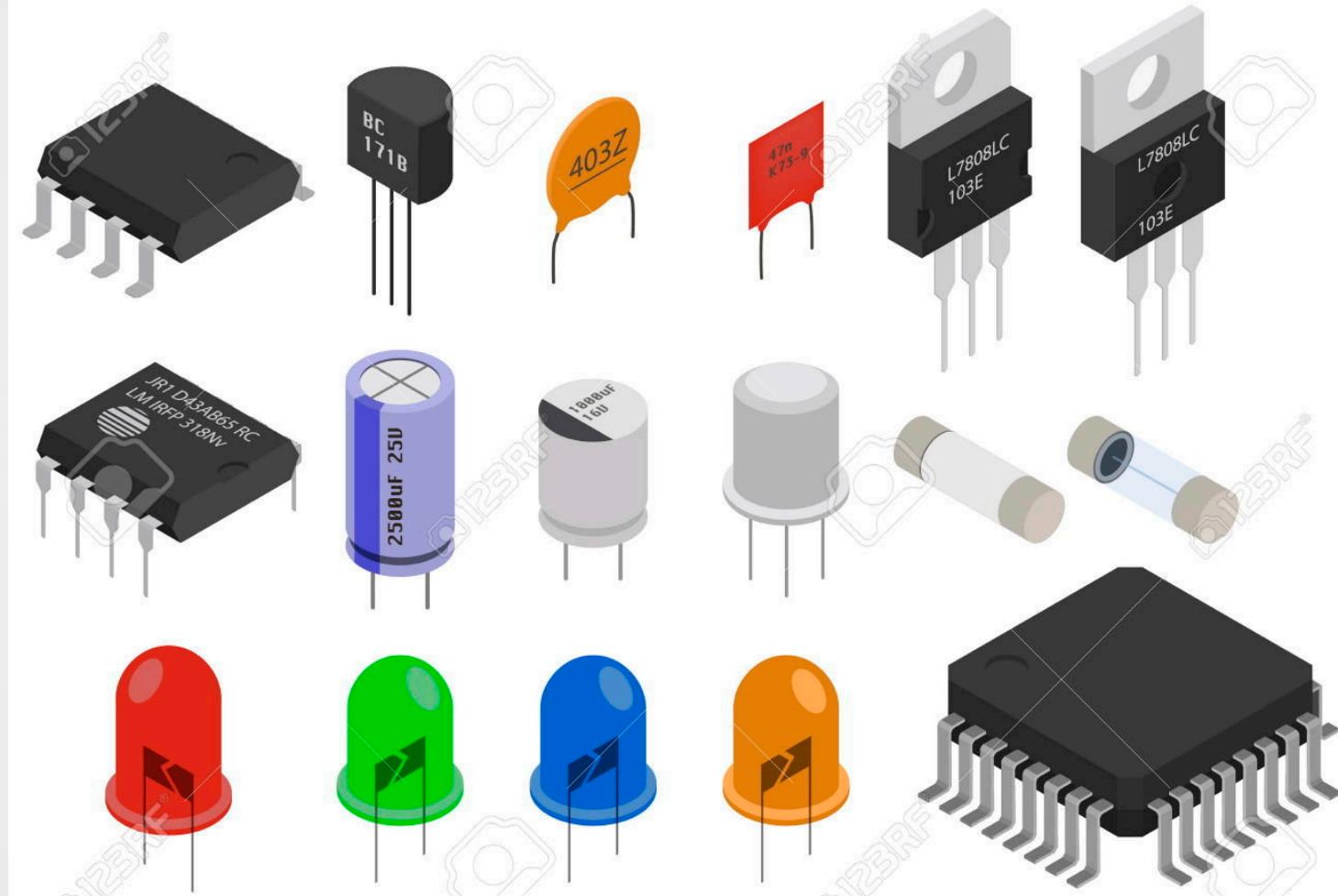


Vuex



Por que preciso disso?

Componentes



Estamos na era dos componentes

Componentes

Algumas características

- Pequenas unidades
- Altamente especializados
- Facilmente testáveis
- Permitem composição
- Fazem com que você DRY
- Encapsulam estrutura, visual e comportamento

Componentes

E principalmente...

TROCAM INFORMAÇÃO ENTRE SI

Componentes

E principalmente...

TROCAM INFORMAÇÃO ENTRE SI

E aqui é que mora a armadilha!

Componentes

Estrutura de um típico componente Vue.js

```
<script>
  export default {
    // COMPORTAMENTO
  }
</script>

<template>
  <div>ESTRUTURA</div>
</template>

<style>
  /* VISUAL */
</style>
```

Componentes

O **estado** local de um componente

```
<script>
  export default {
    data () {
      return {
        list: [1, 2, 3]
      }
    }
  }
</script>
```

Componentes

State

```
{  
  list: ['Eletrônicos', 'Utilidades Domésticas', 'Produtos de Limpeza'],  
  label: 'Categoria selecionada',  
  filters: {  
    date: '08/07/2017',  
    region: 'SP',  
    acknowledgement: false  
  }  
}
```

Reatividade

Shared State

Componente 1 Componente 6 Componente 37

```
{  
    // chamada AJAX abastece o array abaixo...  
  
    list: ['Eletrônicos', 'Utilidades Domésticas', 'Produtos de Limpeza']  
}
```

Shared State

O que fazer para compartilhar este pedaço do **state** ?

- Duplicar a informação?
- NÃO!!!
- Enviar a informação disparando um evento?
- TALVEZ!
- Manter um state **isolado de componentes**, no **nível da aplicação** e **disponível** para **qualquer componente** que tenha interesse na informação?
- YES!!!!

Vuex

colocando de forma simples:

*"Um state isolado de componentes,
no nível da aplicação e disponível
para qualquer componente que tenha interesse
na informação."*

Vuex

- Um pattern e uma library
- Desenvolvido pelo core team Vue
- No fundo apenas um objeto Javascript
- Disponível para todos os componentes
- Alterações automaticamente propagadas
- Por isso **Single Source of Truth**

Vuex



Vuex

- Vuex possui uma forma específica de lidar com a informação
- Você precisará aprender o jeito Vuex
- O desenvolvimento ficará mais burocrático

Instalação e Configuração

Instale com NPM ou Yarn

```
npm install vuex --save  
// ou  
yarn add vuex
```

Instalação e Configuração

Configure no main.js

```
import Vue from 'vue'  
import Vuex from 'vuex'  
import App from './App.vue'  
  
Vue.use(Vuex)  
  
const store = new Vuex.Store({  
  state: {  
    categories: [1, 2, 3]  
  }  
})  
  
new Vue({  
  el: '#app',  
  store, // alias para store: store  
  render: h => h(App)  
})
```

Instalação e Configuração

```
<script>
  export default {
    computed: {
      categories () {
        return this.$store.state.categories
      }
    }
  }
</script>
```

Vuex Store

Um resumo

- Contém os dados: **State**
- Provê métodos para gerenciar o State:

Mutations, Actions e Getters

- Pode ser modularizada: **Modules**

Peculiaridades

- Apenas a Store pode modificar o State
- Fluxo unidirecional de informação
- Store provê todos os métodos necessários para gerenciar o State

Peculiaridades

Apenas a Store pode modificar o state

```
<script>
  export default {
    computed: {
      categories () {
        return this.$store.state.categories
      }
    },
    methods: {
      add (category) {
        this.$store.state.categories.push(category) // < isso NÃO pode!
      }
    }
  }
</script>
```

ERRADO

Peculiaridades

Apenas a Store pode modificar o State

```
<script>
  export default {
    computed: {
      categories () {
        return this.$store.state.categories
      }
    },
    methods: {
      add (category) {
        this.$store.dispatch('add_category', category)
      }
    }
  }
</script>
```

CERTO

Apenas a Store pode modificar o State

add_category é um método de Store cujo tipo, na nomenclatura do pattern Vuex, é **Action**.

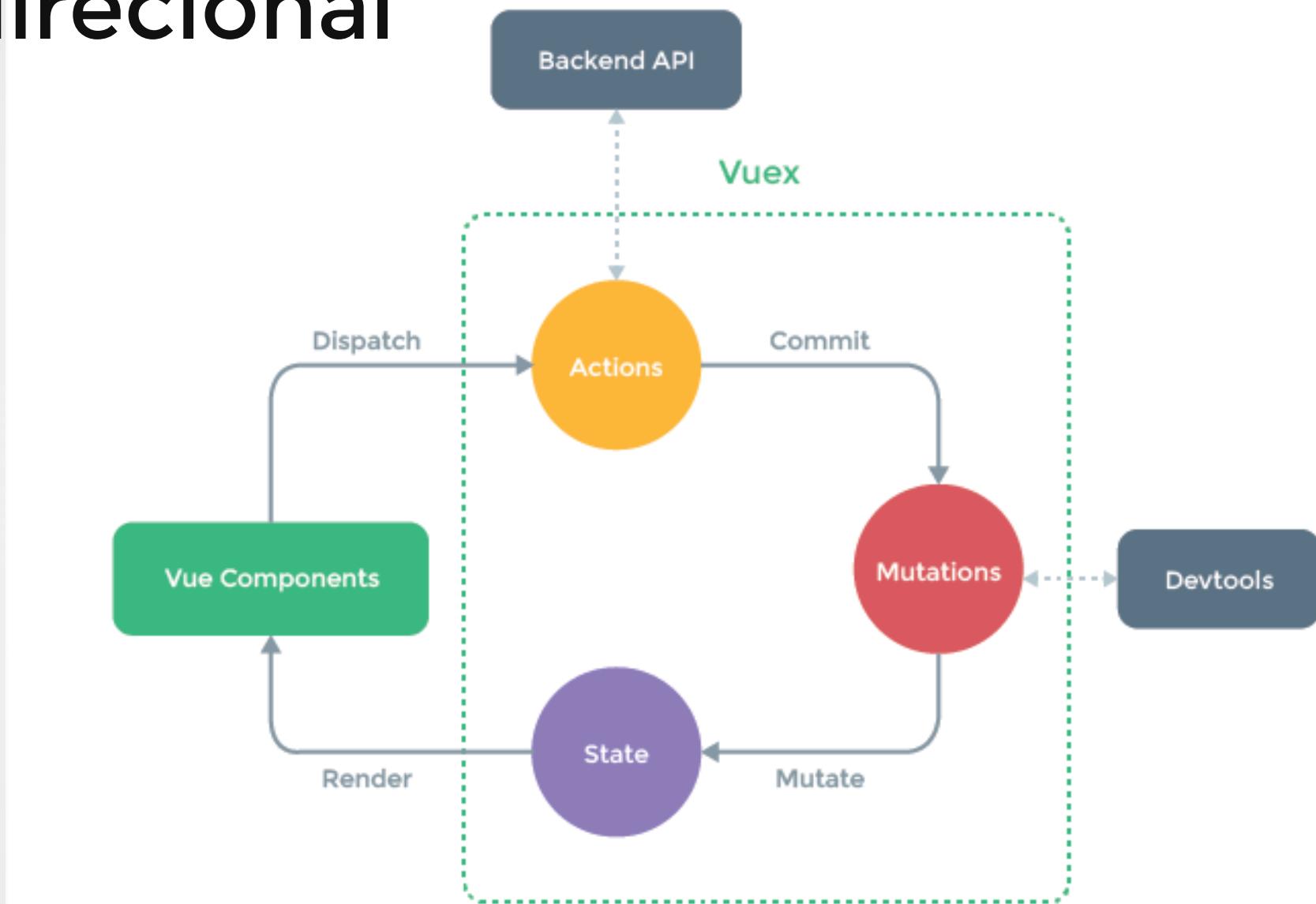
Actions são responsáveis por executar métodos do tipo **Mutation** e *apenas uma Mutation pode alterar o State*.

Actions e Mutations

```
const store = new Vuex.Store({
  state: {
    categories: [1, 2, 3]
  },
  mutations: {
    ADD_CATEGORY (state, obj) { // < variavel state injetada pelo Vuex
      state.categories.push(obj)
    }
  },
  actions: {
    add_category ({commit}, obj) {
      commit('ADD_CATEGORY', obj)
    }
  }
})
```

```
new Vue({
  el: '#app',
  store, // alias para store: store
  render: h => h(App)
})
```

Fluxo Unidirecional



Complicado?

Poderia ser pior!



Demo

Finalizando

- Sou obrigado a usar Vuex?
- Como saber quando preciso?
- Terei que armazenar todos os dados na Store Vuex?



FÁBIO VEDOVELLI

Head of Front End Development at octimine Technologies
Munich, Germany

- . web developer
- . php e javascript
- . laravel e vue.js

Twitter: [@vedovelli](https://twitter.com/vedovelli)

