探索Vue的业务模块化设计

Progressive Enhancement Architecture

Michael Lin 林长青

RingCentral Senior Frontend Engineer

Open Source Repositories

- USM: Universal state module library
- Tees: Universal test framework for front-end

Github

https://github.com/unadlib

Table of Contents

- Motivation
- State Model & Vuex Modularity Design
- Modular Classification & Modular Elements
- Modular Engineering
- Demo
- Q&A

Motivation

Vue on large scale system

Modularity

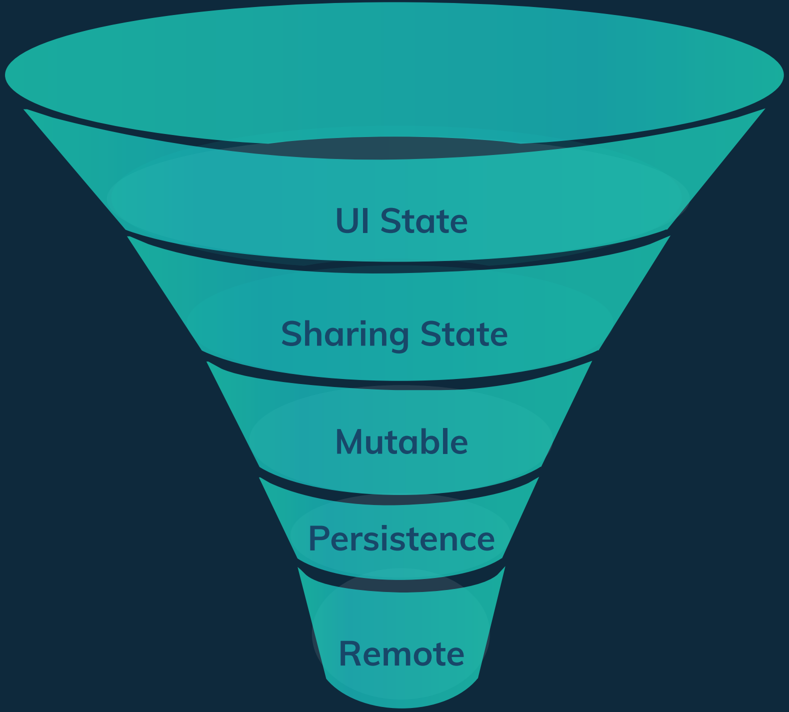
Core Business Logic Separation from UI

Divide & Conquer

Test Automation & Product Iteration

State Model &
Vuex Modularity Design

State Model



UI State

Sharing State

Mutable

Persistence

Remote

1. UI state

Mixins / Custom Directives / Plugins / Filters

Renderless Components / HOC

Function-based Component

Vue-hooks

2. Sharing State

Vuex Modularity

2.1
Vuex State Management Modularity

Modular State

- OOP
- Module injection

Optional Life Cycle

- Initialization
- Module unmount

Optional Event System

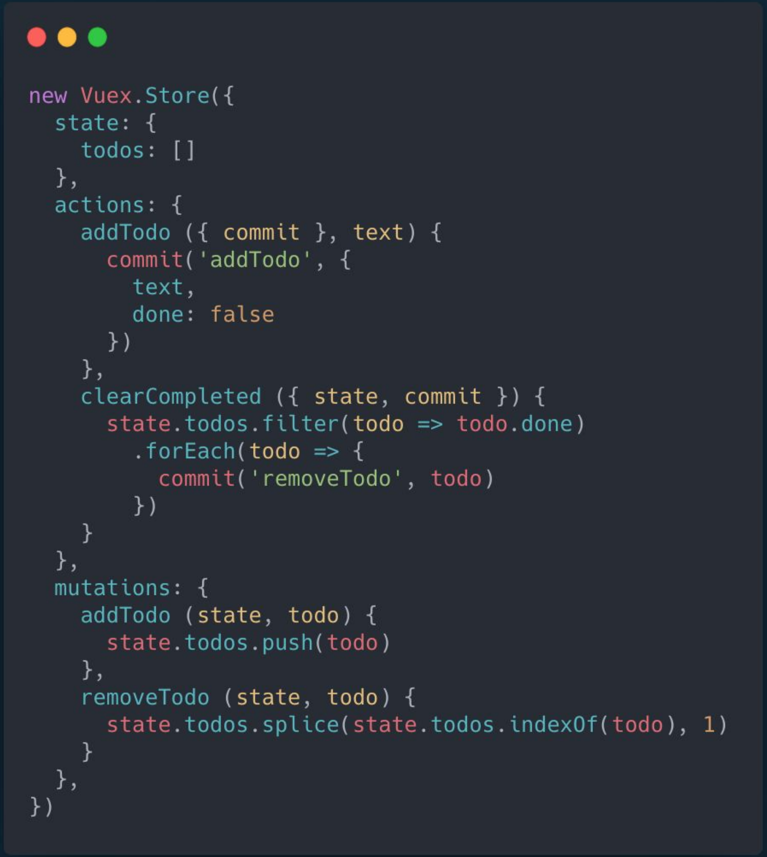
- Modules interaction
- Message Transfer

Solution

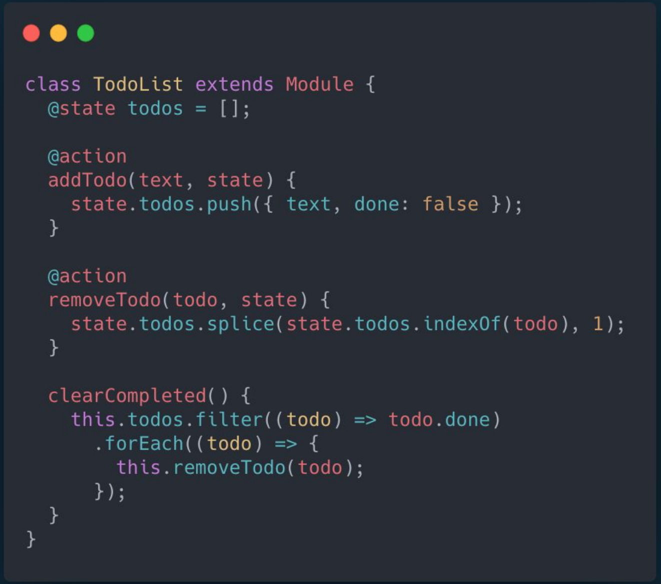
usm-vuex

2.2 Vuex Modularity

Class-based Module

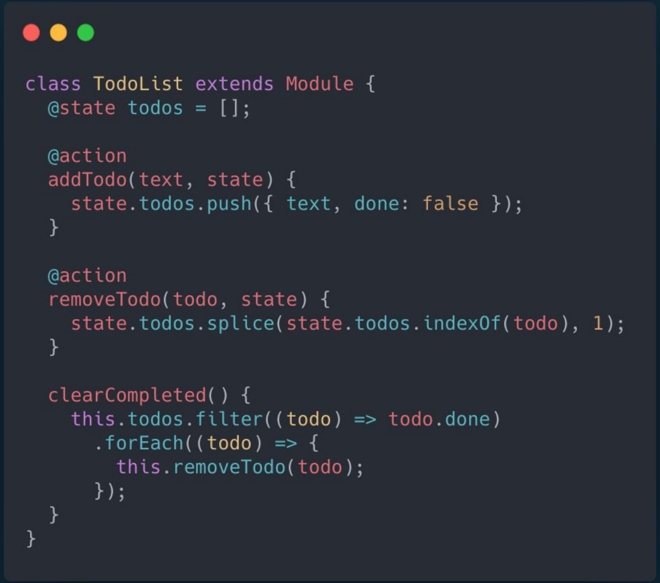


```
new Vuex.Store({
  state: {
    todos: []
  },
  actions: {
    addTodo ({ commit }, text) {
      commit('addTodo', {
        text,
        done: false
      })
    },
    clearCompleted ({ state, commit }) {
      state.todos.filter(todo => todo.done)
        .forEach(todo => {
          commit('removeTodo', todo)
        })
    }
  },
  mutations: {
    addTodo (state, todo) {
      state.todos.push(todo)
    },
    removeTodo (state, todo) {
      state.todos.splice(state.todos.indexOf(todo), 1)
    }
  },
})
```

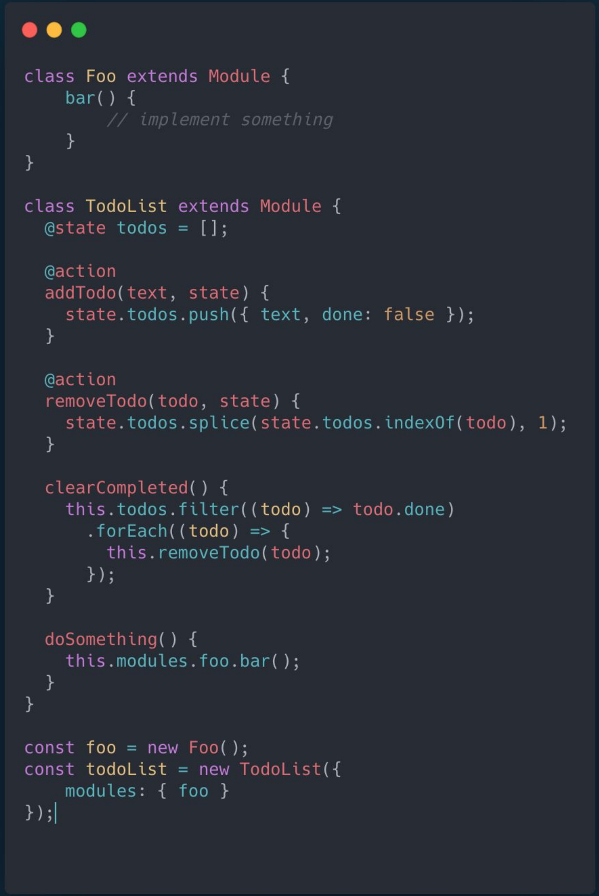


```
class TodoList extends Module {
  @state todos = [];

  @action
  addTodo(text, state) {
    state.todos.push({ text, done: false });
  }

  @action
  removeTodo(todo, state) {
    state.todos.splice(state.todos.indexOf(todo), 1);
  }

  clearCompleted() {
    this.todos.filter((todo) => todo.done)
      .forEach((todo) => {
        this.removeTodo(todo);
      });
  }
}
```

2.3 DI/OO

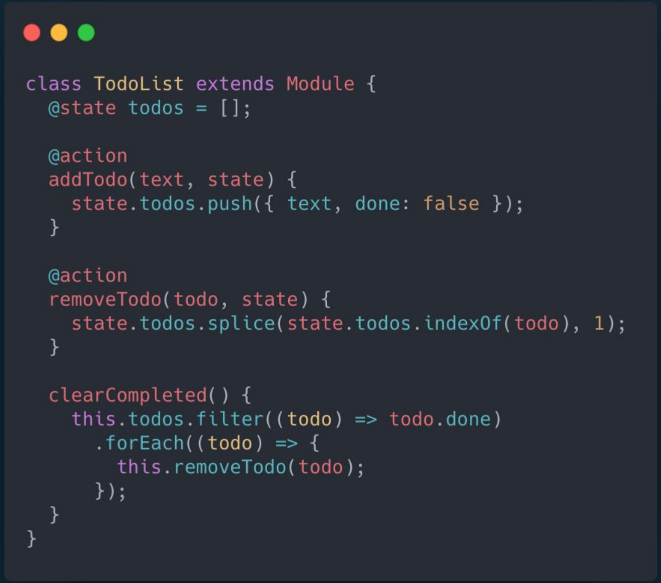


```
class TodoList extends Module {
  @state todos = [];

  @action
  addTodo(text, state) {
    state.todos.push({ text, done: false });
  }

  @action
  removeTodo(todo, state) {
    state.todos.splice(state.todos.indexOf(todo), 1);
  }

  clearCompleted() {
    this.todos.filter((todo) => todo.done)
      .forEach((todo) => {
        this.removeTodo(todo);
      });
  }
}
```



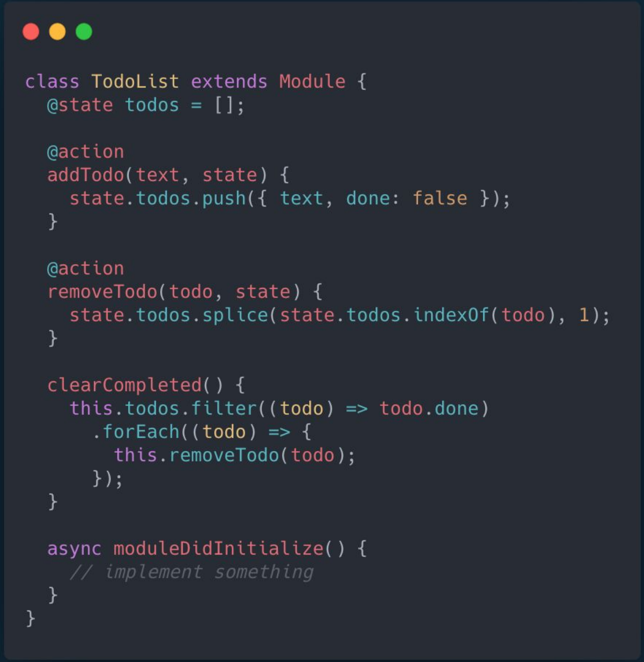
```
class Foo extends Module {
    bar() {
        // implement something
    }
}

class TodoList extends Module {
  @state todos = [];

  @action
  addTodo(text, state) {
    state.todos.push({ text, done: false });
  }

  @action
  removeTodo(todo, state) {
    state.todos.splice(state.todos.indexOf(todo), 1);
  }

  clearCompleted() {
    this.todos.filter((todo) => todo.done)
      .forEach((todo) => {
        this.removeTodo(todo);
      });
  }

  doSomething() {
    this.modules.foo.bar();
  }
}

const foo = new Foo();
const todoList = new TodoList({
    modules: { foo }
});
```

2.4 Optional Life Cycle



```
class TodoList extends Module {
  @state todos = [];

  @action
  addTodo(text, state) {
    state.todos.push({ text, done: false });
  }

  @action
  removeTodo(todo, state) {
    state.todos.splice(state.todos.indexOf(todo), 1);
  }

  clearCompleted() {
    this.todos.filter((todo) => todo.done)
      .forEach((todo) => {
        this.removeTodo(todo);
      });
  }
}
```

→



```
class TodoList extends Module {
  @state todos = [];

  @action
  addTodo(text, state) {
    state.todos.push({ text, done: false });
  }

  @action
  removeTodo(todo, state) {
    state.todos.splice(state.todos.indexOf(todo), 1);
  }

  clearCompleted() {
    this.todos.filter((todo) => todo.done)
      .forEach((todo) => {
        this.removeTodo(todo);
      });
  }

  async moduleDidInitialize() {
    // implement something
  }
}
```

2.5 Optional Event System

- EventEmitter
- RxJS
- Custom Event

3. Mutable

- Services Layer
- Universal Layer
- High-performance Layer

WebAssembly / Worker, etc.

4.Persistence

Persistence

- ORM (vuex-orm/type-orm)
- Storage wrapper (localforage/dexie)

5. Remote

Remote Data

- Request Module / SDK
- RESTful / GraphQL

Data parser(Normalizr, etc.)

Modular Classification & Modular Elements

Module Classification

By Type

- Router Module
- High Performance Module
- Business Logic Module
- UI Logic Module
- External Module
- Other Basic Module

By Usage

- View Module
- Service Module

Modular Elements

- Dependency Management
- Package Management
- Event System
- Lazy Loading
- General Logic
- Component-based Design

1

Dependency Management

- Dependency Injection
- Service Locator
- Other IoC (inversify, etc.)

2

Event System

- Eventemitter
- RxJS
- Custom Event

3

General Logic

- Domain-Driven Design
- Principles(SOLID / KISS / DRY, etc.)

4

Package Management

- Multi-repo / Monorepo
- Lerna / oao, etc.

5

Lazy Loading

- Modules and UIs of the subpackage
- DI Asynchronous Factory
- Vuex register Module
- Dynamic Import
- Module Loader(Requirejs..)

6

Component-based Design

- Mixins design
- Function-based Component

Modular Engineering

Engineering Monorepo/ Multi-repo / Tools / Business Logic Packages, etc.

Automation CI/CD/AT, etc.

Development Mode Workflow/Branch Model(Branch by Abstraction), etc.

Programming Pattern OOP/FP/RFP, etc.

Model Visualization Flow Chart, etc.

DEMO

https://github.com/unadlib/usm-vuex-demo

So why not use Angular?

- GUI State Model: Mutable < Immutable < Observable
- Flexible(Better IoC, etc.)
- Customizable, Optional
- Progressive Enhancement

Q & A

Thanks!

https://github.com/unadlib/usm

You can find me at:

unadlib@gmail.com