复习

核心api

组件设计

- input
  - 双向绑定

    @input，:value
  - 传值prop
  - 事件

    this.$emit()

    this.$parent.$emit()
- form-item
  - 插槽 slot
  - 事件

    this.$on()
  - 注入 inject
- form
  - 插槽 slot
  - 隔层传值 provide

作业

1. 不同事件触发方式
2. 自定义校验
3. 隔层事件派发

今日知识

- vue-router
  - 安装：`vue add router`
  - 路由配置

    export default new VueRouter({ routes: [{ path: '/', name: 'home', component: Home }] })
  - 导航

    Home
  - 路由出口
  - 嵌套
    - 配置：children

{ path: "/",
component: Home,
children: [{ path: "", name: "home", component: List }] }

- 父组件：

- 传参

  { path: "detail/:id", component: Detail }

  {{$route.params.id}}

- 路由守卫
  - 全局beforeEach(to,from,next)
  - 路由独享beforeEnter(to,from,next)
  - 组件beforeRouteEnter(to,from,next)

- vuex
  - 安装 vue add vuex
  - 状态与变更

    export default new Vuex.Store({ state: { count:0 }, mutations: { add(state) { state.count++ } } })

  - 派生状态

    export default new Vuex.Store({ getters: { todoCount(state) { return state.todos.filter(todo=>!todo.completed).length } } })

  - 异步操作

    export default new Vuex.Store({ actions: { someAction(context) { // do something context.state;// 访问状态 context.commit();// 提交变更 context.dispatch();// 派发动作 } } })

  - 简化方法

    export default { computed: { ...mapState(['isLogin']), ...mapGetters(['loginState']) },

    methods: { ...mapActions(['login']) }

    }

- 原理
  - vuex

```
class KStore {
  constructor(options) {
    this.state = options.state;
    this.mutations = options.mutations;
    this.actions = options.actions;
    // 借用vue本身的数据响应式机制
    this.vm = new Vue({
      data: {
        state: this.state
      }
    });
  }

  commit(type, payload) {
```

```
    const mutation = this.mutations[type];
    mutation(this.state, payload);
  }

  dispatch(type, payload) {
    const action = this.actions[type];
    const ctx = {
        commit: this.commit,
        state: this.state,
        dispatch: this.dispatch
    }
    return action(ctx, payload);
  }
}
```

- router

```
class VueRouter {
    constructor(Vue, options) {
        this.$options = options
        this.routeMap = {}
        this.app = new Vue({
            data: {
                current: '#/'
            }
        })

        this.init()
        this.createRouteMap(this.$options)
        this.initComponent(Vue)
    }

    // 初始化 hashchange
    init() {
        window.addEventListener('load', this.onHashChange.bind(this), false)
        window.addEventListener('hashchange', this.onHashChange.bind(this),
false)
    }

    createRouteMap(options) {
        options.routes.forEach(item => {
            this.routeMap[item.path] = item.component
        })
    }

    // 注册组件
    initComponent(Vue) {
        Vue.component('router-link', {
            props: {
                to: String
            },
```

```
            render: function(h) {
                // h <==> createElement
                // <a :href="to"><slot></slot></a>
                return h(
                    "a",
                    { attrs: { href: this.to } },
                    this.$slots.default
                );
            }
        })

        const _this = this
        Vue.component('router-view', {
            render(h) {
                var component = _this.routeMap[_this.app.current]
                return h(component)
            }
        })
    }

    // 获取当前 hash 串
    getHash() {
        return window.location.hash.slice(1) || '/'
    }

    // 设置当前路径
    onHashChange() {
        this.app.current = this.getHash()
    }
}
```