Vue-admin-temeplate

拓展知识点

页面级别权限

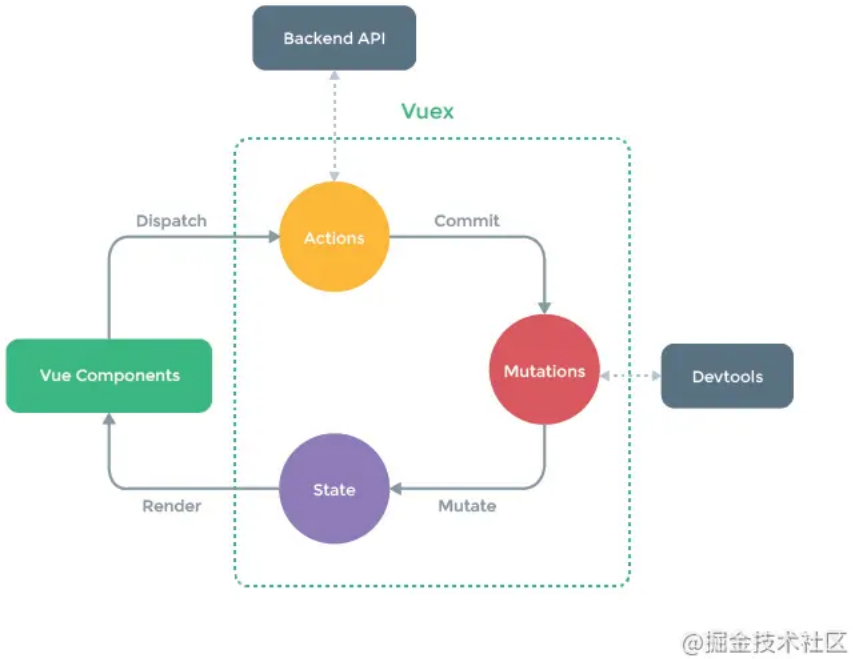
```
path: '/permission',
component: Layout,
redirect: '/permission/page',
alwaysShow: true, // will always show the root menu
name: 'Permission',
meta: {
  title: 'Permission',
  icon: 'lock',
  roles: ['admin', 'editor'] // you can set roles in root nav
},
children: [
  {
    path: 'page',
    component: () => import('@/views/permission/page'),
    name: 'PagePermission',
    meta: {
      title: 'Page Permission',
      roles: ['admin'] // or you can only set roles in sub nav
    }      Pan, 3 years ago • format code
  },
  {
    path: 'directive',
    component: () => import('@/views/permission/directive'),
    name: 'DirectivePermission',
    meta: {
      title: 'Directive Permission'
      // if do not set roles, means: this page does not require permission
    }
  },
```

按钮级别权限

```
<div :key="key" style="margin-top:30px;">
  <div>      Pan, 3 years ago • perf[v-permission]: refine v-permission demo
    <span v-permission="['admin']" class="permission-alert">
      Only
      <el-tag class="permission-tag" size="small">admin</el-tag> can see this
    </span>
    <el-tag v-permission="['admin']" class="permission-sourceCode" type="info">
      v-permission="['admin']"
    </el-tag>
  </div>

  <div>
    <span v-permission="['editor']" class="permission-alert">
      Only
      <el-tag class="permission-tag" size="small">editor</el-tag> can see this
    </span>
    <el-tag v-permission="['editor']" class="permission-sourceCode" type="info">
      v-permission="['editor']"
    </el-tag>
  </div>
```

VueX原理图


Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools
State
Render
Mutate
@掘金技术社区

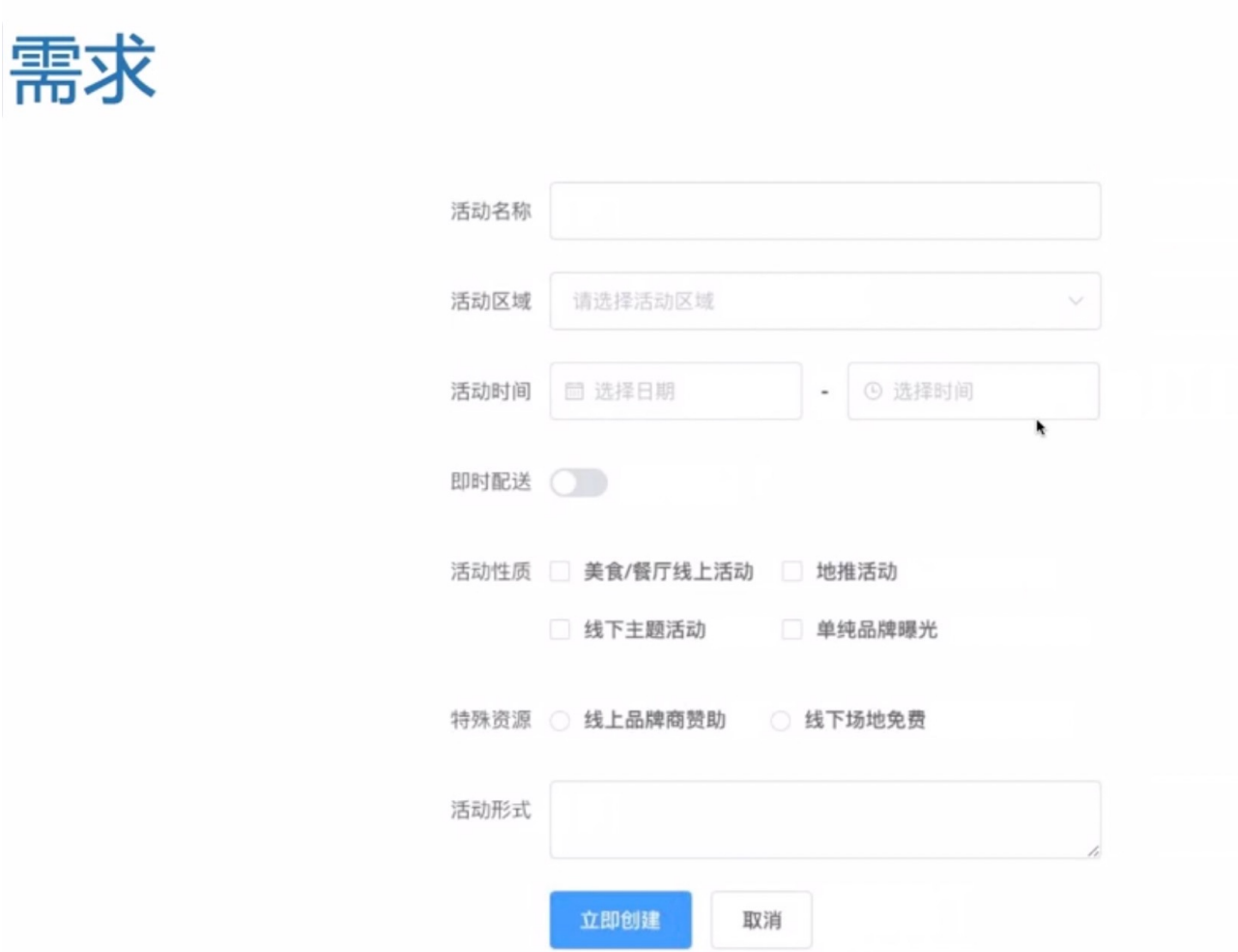
内存使用

performance的memory属性对象

```
memory: {
        jsHeapSizeLimit: 2172649472 // 内存大小限制
        totalJSHeapSize: 5891850 // 可使用的内存
        usedJSHeapSize: 4044814 // JS 对象（包括V8引擎内部对象）占用的内存，一定小于 totalJSHeapSize
}
```

案例需求

需求


活动名称
活动区域
请选择活动区域
活动时间
选择日期
-
选择时间
即时配送
活动性质
美食/餐厅线上活动
地推活动
线下主题活动
单纯品牌曝光
特殊资源
线上品牌商赞助
线下场地免费
活动形式
立即创建
取消

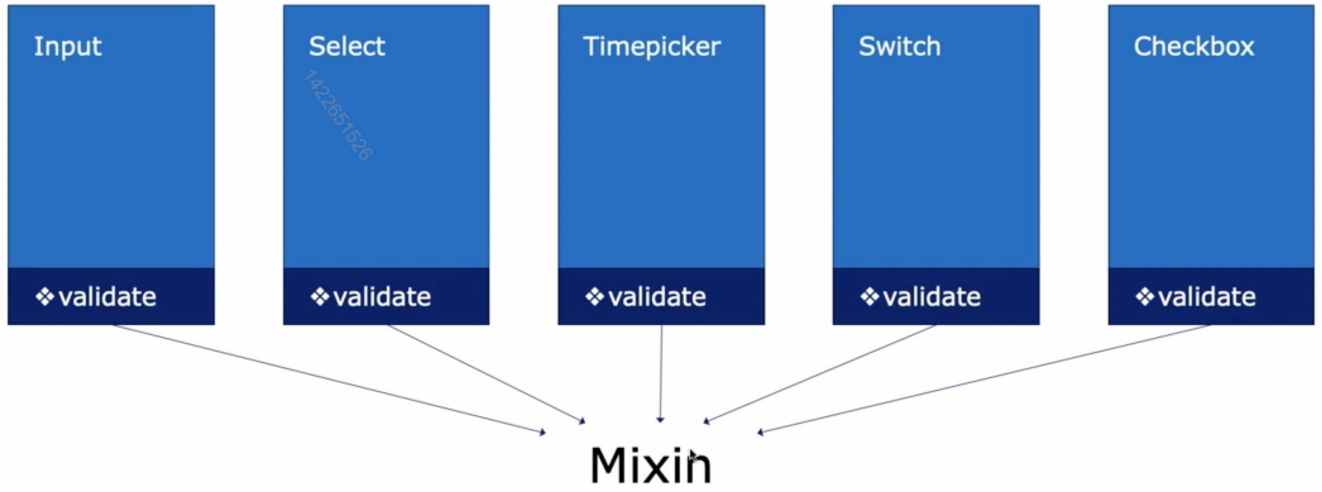
组件复用

Mixins

HOC

Renderless 组件

Mixin


Input
validate
Select
validate
Timepicker
validate
Switch
validate
Checkbox
validate
Mixin

Mixin

```vue
<template>
  <div>
    <input type="text" @blur="blur" />
    {{ errmsg }}
  </div>
</template>

<script>
import validateMixin from "./mixin";

export default {
  mixins: [validateMixin],
  data: () => ({ errmsg: "" }),
  methods: {
    blur() {
      this.validate();
    }
  }
};
</script>
```

缺陷

validateMixin

```vue
export default {
  methods: {
    validate() {
      /**
       * 校验逻辑
       */

      return true;
    }
  }
};
```

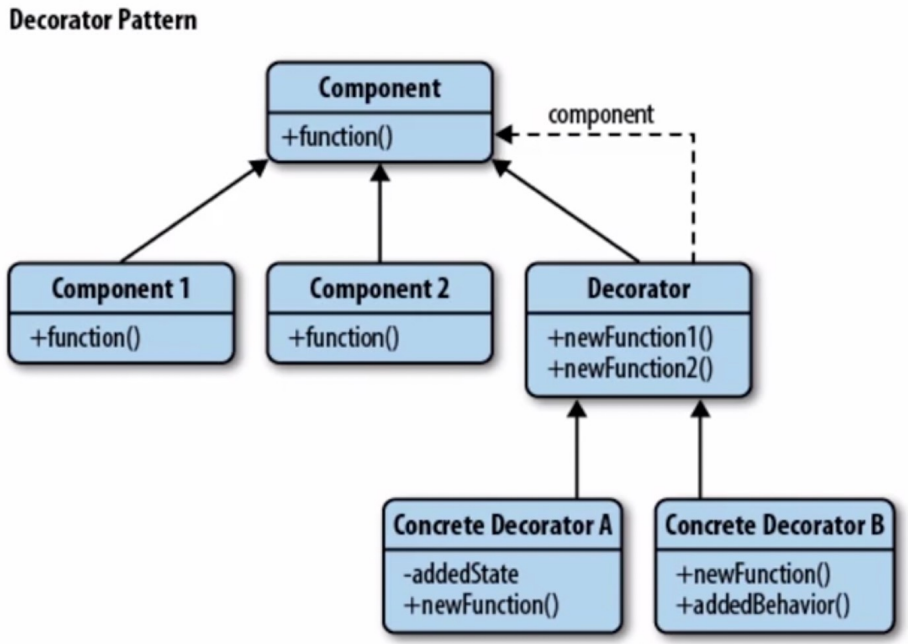
- 同名钩子函数将合并为一个数组，混入对象的钩子将在组件自身钩子之前调用。
- 二者的 methods、components 和 directives，将被合并为同一个对象。若对象键名冲突时，取组件对象的键值对。

Mixin

缺陷

- 打破了原有组件的封装
- 增加组件复杂度
- 可能会出现命名冲突的问题
- 仅仅只是对逻辑的复用，模板不能复用

HOC



HOC: 函数接收一个组件作为参数，并返回一个新组件； 可复用的逻辑在函数中实现

HOC

HOC组件

```
import Vue from "vue";
const ValidateHoc = Component => {
  return Vue.component(`hoc-${Component.name}`, {
    data: () => ({ errMsg: "" }),
    methods: {
      validate() {
        // eslint-disable-next-line no-console
        console.log("validate");
        /**
         * 校验逻辑
         */

        return true;
      }
    },
    render() {
      return (
        <div>
          <Component on-blur={this.validate} />
          {this.errMsg}
        </div>
      );
    }
  });
};
export default ValidateHoc;

```

input组件

```
<template>
  <input type="text" @blur="$emit('blur')" />
</template>
```

外层组件（组装HOC组件和input组件）

```
import CustomInput from "./components/composition/2/CustomInput";
import ValidateHoc from "./components/composition/2/Hoc.js";

const ValidateInput = ValidateHoc(CustomInput);

export default {
  name: "app",
  render() {
    return <ValidateInput />;
  }
};
```

HOC

相比较Mixin的优点：

- 模板可复用
- 不会出现命名冲突（本质上是一个HOC是套了一层父组件）

不足：

- 组件复杂度高，多层嵌套，调试会很痛苦

RenderLess

- 复用的逻辑沉淀在包含slot插槽的组件
- 接口由插槽Prop来暴露

```vue
<template>
  <s-validate #default="{ validate }" :value="value" :rules="rules">
    <input type="text" @blur="validate" v-model="value" />
  </s-validate>
</template>

<script>
import SValidate from "./SValidate";

export default {
  data: () => ({ value: "hi", rules: [] }),
  components: {
    SValidate
  }
};
</script>
```

SValidate.vue

```vue
<template>
  <div>
    <slot :validate="validate"></slot>
    {{ errMsg }}
  </div>
</template>
<script>
export default {
  props: ["value", "rules"],
  data() {
    return { errMsg: "" };
  },
  methods: {
    validate() {
      let validate = this.rules.reduce((pre, cur) => {
        return pre && cur && cur.test && cur.test(this.value);
      }, true);
      return validate;
    }
  }
};
</script>
```

RenderLess

优点：

- 模板可复用
- 不会出现命名冲突
- 符合依赖倒置原则
- 复用的接口来源清晰