

Vue组件设计与实践

第三方组件应用

组件设计：实现Form、FormItem、Input

实现Input：

实现FormItem

实现Form：

数据校验

第三方组件应用

- Element-UI: <http://element-cn.eleme.io/>
- element集成: vue add element
- 组件使用: 创建一个登陆表单并可以校验用户输入

```
<template>
  <div>
    <h3>Element表单</h3>
    <hr>
    <el-form :model="model" :rules="rules" ref="loginForm">
      <el-form-item label="用户名" prop="username">
        <el-input v-model="model.username" autocomplete="off"></el-input>
      </el-form-item>
      <el-form-item label="确认密码" prop="password">
        <el-input type="password" v-model="model.password" autocomplete="off"></el-
input>
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click="submitForm('loginForm')>提交</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>

<script>
export default {
  data() {
    return {
      model: { username: "tom", password: "" },
      rules: {
        username: [{ required: true, message: "请输入用户名" }],
        password: [{ required: true, message: "请输入密码" }],
      }
    }
  },
},
```

```

methods: {
  submitForm(form) {
    this.$refs[form].validate(valid=>{
      if (valid) {
        alert('请求登录!')
      } else {
        alert('校验失败! ')
      }
    })
  },
};
</script>

```

> 需要在element.js导入Form、FormItem和Input

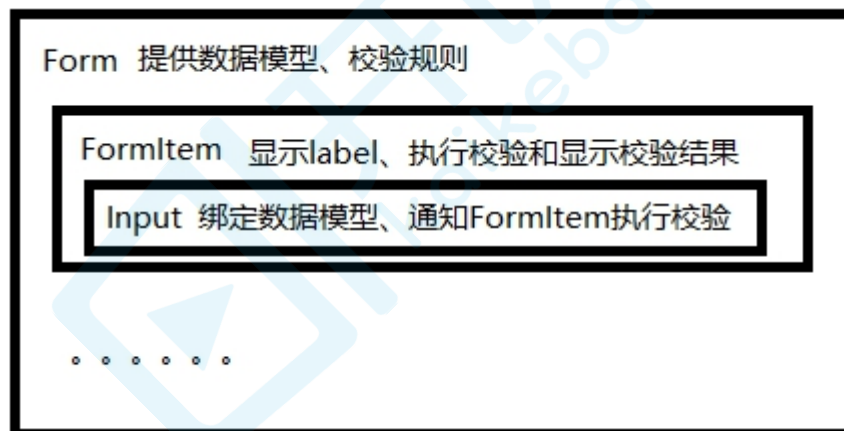
```
import { Button,Form,FormItem,Input } from 'element-ui'
```

```
Vue.use(Form)
```

```
Vue.use(FormItem)
```

```
Vue.use(Input)
```

组件设计：实现Form、FormItem、Input



实现Input:

```

<template>
  <div>
    <input :type="type" :value="inputValue" @input="onInput">
  </div>
</template>

<script>
  export default {
    props: {
      value: {
        type: String,

```

```

        default: ''
      },
      type: {
        type: String,
        default: 'text'
      }
    },
    data() {
      return { // 数据流是单向的, input的值应该是内部值inputValue而非属性value
        inputValue: this.value
      }
    },
    watch: { // 把属性value变化同步给inputValue
      value(newValue) {
        this.inputValue = newValue;
      }
    },
    methods: { // input事件触发设置模型的值并通知父组件
      onInput(e) {
        this.inputValue = e.target.value;
        this.$emit('input', this.inputValue);
      }
    },
  },
}
</script>

```

实现FormItem

```

<template>
  <div>
    <label v-if="label">{{label}}</label>
    <slot></slot>
    <p v-if="error">{{error}}</p>
  </div>
</template>

<script>
export default {
  props: {
    label: { // 输入项标签
      type: String,
      default: ''
    },
    prop: { // 字段名
      type: String,
      default: ''
    },
  },
  data() {
    return {

```

```
      error: '' // 校验错误
    },
  },
};
</script>
```

实现Form:

```
<template>
  <form>
    <slot></slot>
  </form>
</template>

<script>
export default {
  provide() {
    return {
      form: this // 将组件实例作为提供者，子代组件可方便获取
    };
  },
  props: {
    model: { type: Object, required: true },
    rules: { type: Object }
  }
};
</script>
```

数据校验

- 通知校验

```
onInput(e) {
  // ...
  // $parent指FormItem
  this.$parent.$emit('validate');
}
```

- FormItem监听校验通知，获取规则并执行校验

```

inject: ['form'], // 注入
mounted(){// 监听校验事件
    this.$on('validate', this.validate)
},
methods: {
    validate() {
        // 获取对应FormItem校验规则
        console.log(this.form.rules[this.prop]);
        // 获取校验值
        console.log(this.form.model[this.prop]);
    }
},

```

- 安装async-validator: npm i async-validator -S

```

import schema from "async-validator";

validate() {
    // 获取对应FormItem校验规则
    const rules = this.form.rules[this.prop];
    // 获取校验值
    const value = this.form.model[this.prop];
    // 校验描述对象
    const descriptor = { [this.prop]: rules };
    // 创建校验器
    const schema = new Schema(descriptor);
    schema.validate({ [this.prop]: value }, errors => {
        if (errors) {
            // 将错误信息显示
            this.error = errors[0].message;
        } else {
            // 校验通过
            this.error = "";
        }
    });
}

```