组件化开发

课程目标

开源组件库

- Element-UI
- iView

CDD

- CDD(Component-Driven Development)
  - 自下而上
  - 从组件级别开始，到页面级别结束

CDD 的好处

- 组件在最大程度被重用
- 并行开发
- 可视化测试

课程介绍

- 处理组件的边界情况
- 快速原型开发
- 组件开发
- Storybook
- Monorepo
- 基于模板生成包的结构
- Lerna + yarn workspaces
- 组件测试
- Rollup 打包

处理组件边界情况

处理组件的边界情况

- $root
- $parent / $children
- $refs

```
this.$refs[formName].validate((valid) => {
  if (valid) {
    alert('submit!');
  } else {
    console.log('error submit!!');
    return false;
  }
});
```

- 依赖注入 provide / inject

$attrs / $listeners

$attrs / $listeners

- $attrs
  - 把父组件中非 prop 属性绑定到内部组件
- $listeners
  - 把父组件中的DOM对象的原生事件绑定到内部组件

快速原型开发

快速原型开发

- VueCLI 中提供了一个插件可以进行原型快速开发
- 需要先额外安装一个全局的扩展
  - npm install -g @vue/cli-service-global
- 使用 vue serve 快速查看组件的运行效果

vue serve

- vue serve 如果不指定参数默认会在当前目录找以下的入口文件
  - main.js、index.js、App.vue 、 app.vue
- 可以指定要加载的组件
  - vue serve ./src/login.vue

快速原型开发

Element-UI

安装 ElementUI

- 初始化 package.json
  - npm init -y
- 安装 ElementUI
  - vue add element
- 加载 ElementUI，使用 Vue.use() 安装插件

组件开发

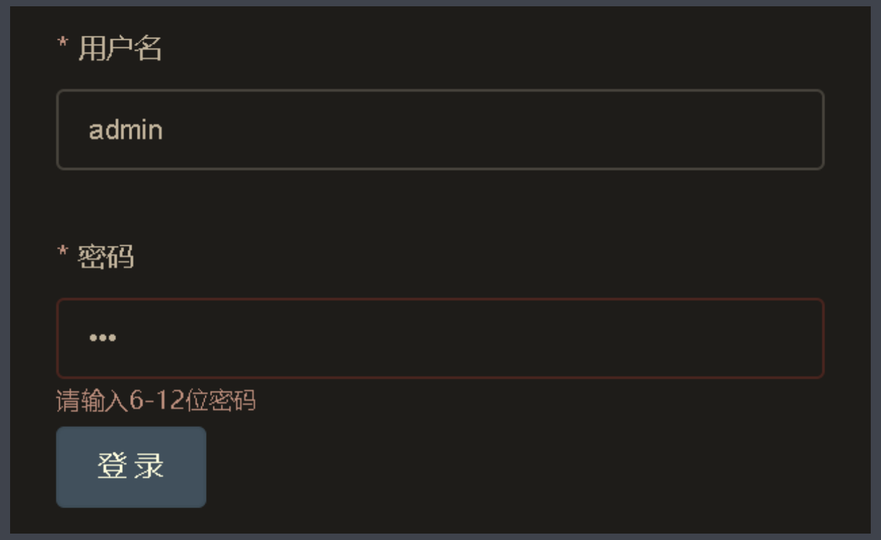
步骤条组件

组件分类

- 第三方组件
- 基础组件
- 业务组件

组件开发

表单组件




```vue
<template>
  <el-form class="form" ref="form" :model="user" :rules="rules">
    <el-form-item label="用户名" prop="username">
      <el-input v-model="user.username"></el-input>
    </el-form-item>
    <el-form-item label="密码" prop="password">
      <el-input type="password" v-model="user.password"></el-input>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" @click="login()">登 录</el-button>
    </el-form-item>
  </el-form>
</template>
```

整体结构

- Form
- FormItem
- Input
- Button

组件开发

表单组件

组件开发

表单组件-表单验证-上

组件开发

表单组件-表单验证-下

Input 组件验证

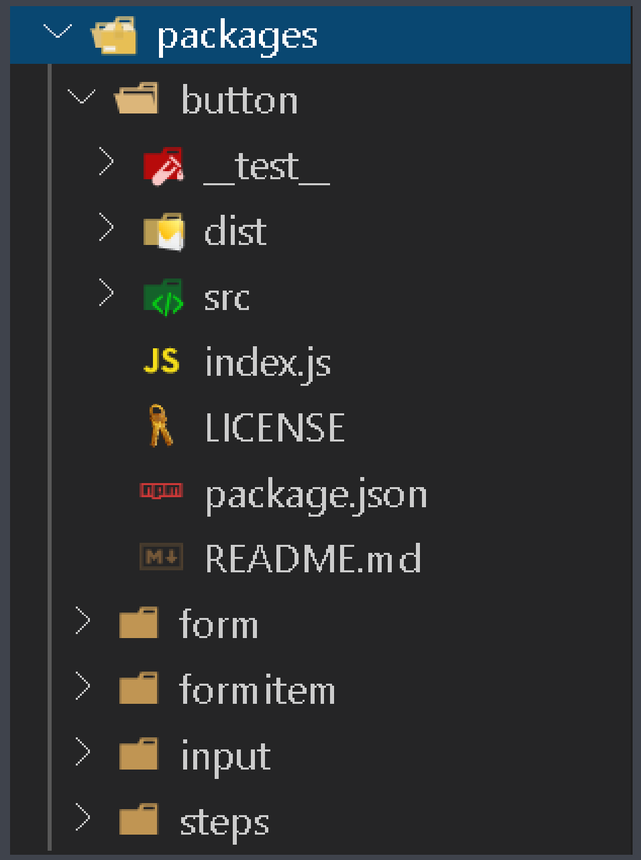
- Input 组件中触发自定义事件 validate
- FormItem 渲染完毕注册自定义事件 validate

Monorepo

两种项目的组织方式

- Multirepo(Multiple Repository)
  - 每一个包对应一个项目
- **Monorepo(Monolithic Repository)**
  - 一个项目仓库中管理多个模块/包

目录结构



Storybook

Storybook

- 可视化的组件展示平台
- 在隔离的开发环境中，以交互式的方式展示组件
- 独立开发组件
- 支持的框架
  - React、React Native、Vue、Angular、
  - Ember、HTML、Svelte、Mithril、Riot

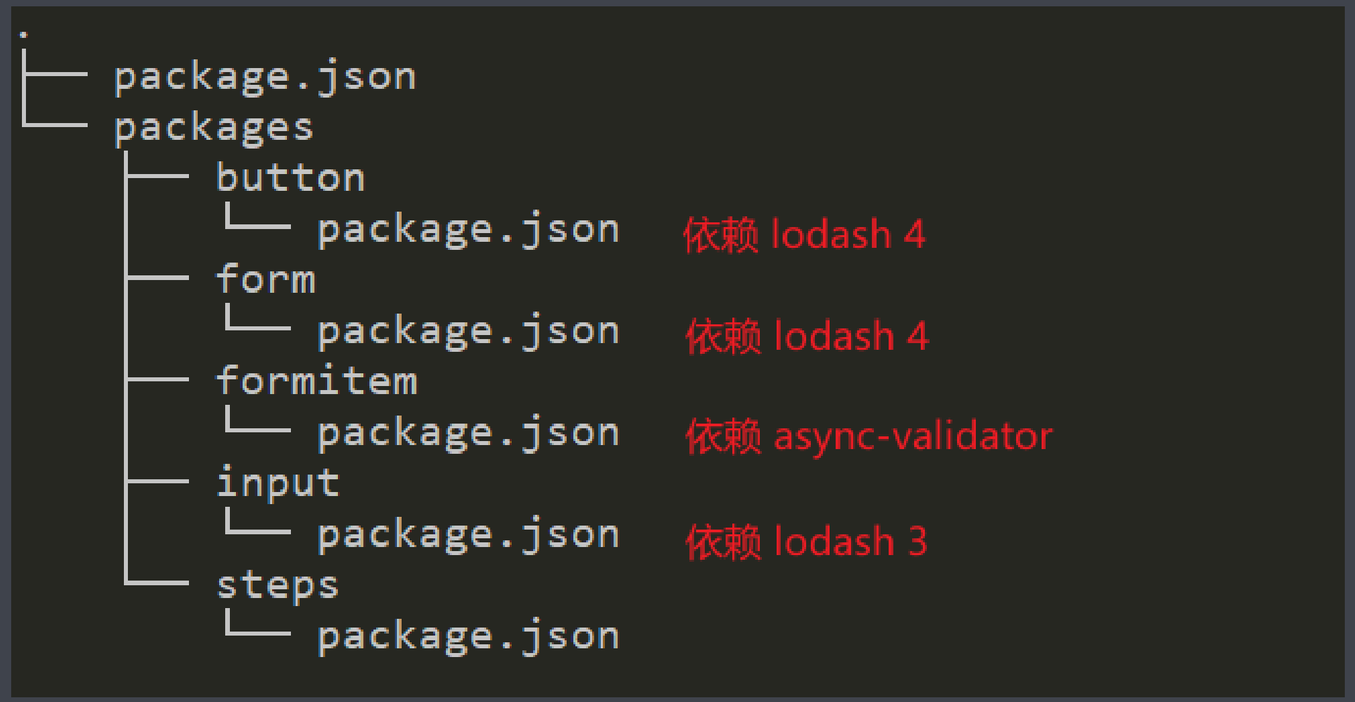
Storybook 安装

- 自动安装
  - npx -p @storybook/cli sb init --type vue
  - yarn add vue
  - vue yarn add vue-loader vue-template-compiler --dev
- 手动安装

Storybook

yarn workspaces

项目依赖


.
package.json
packages
button
package.json
依赖 lodash 4
form
package.json
依赖 lodash 4
formitem
package.json
依赖 async-validator
input
package.json
依赖 lodash 3
steps
package.json

开启 yarn 的工作区

- 项目根目录的 package.json

```
"private": true,
"workspaces": [
  "./packages/*"
]
```

yarn workspaces 使用

- 给工作区根目录安装开发依赖
  - yarn add jest -D –W
- 给指定工作区安装依赖
  - yarn workspace lg-button add lodash@4
- 给所有的工作区安装依赖
  - yarn install

Lerna

Lerna 介绍

- Lerna 是一个优化使用 git 和 npm 管理多包仓库的工作流工具
- 用于管理具有多个包的 JavaScript 项目
- 它可以一键把代码提交到git和npm仓库

Lerna 使用

- 全局安装
  - yarn global add lerna
- 初始化
  - lerna init
- 发布
  - lerna publish

Vue 组件的单元测试

组件单元测试好处

- 提供描述组件行为的文档
- 节省手动测试的时间
- 减少研发新特性时产生的 bug
- 改进设计
- 促进重构

安装依赖

- Vue Test Utils
- Jest
- vue-jest
- babel-jest
- 安装
  - yarn add jest @vue/test-utils vue-jest babel-jest -D -W

配置测试脚本

- package.json

```
"scripts": {
    "test": "jest",
    ……
}
```

Jest 配置文件

- jest.config.js

```
module.exports = {
  "testMatch": ["**/__tests__/**/*.[jt]s?(x)"],
  "moduleFileExtensions": [
    "js",
    "json",
    // 告诉 Jest 处理 `*.vue` 文件
    "vue"
  ],
  "transform": {
    // 用 `vue-jest` 处理 `*.vue` 文件
    ".*\\.(vue)$": "vue-jest",
    // 用 `babel-jest` 处理 js
    ".*\\.(js)$": "babel-jest"
  }
}
```

Babel 配置文件

- babel.config.js

```
module.exports = {
  presets: [
    [
      '@babel/preset-env'
    ]
  ]
}
```

Babel 桥接

- yarn add babel-core@bridge -D -W

Vue 组件的单元测试

Jest 常用 API

- 全局函数
  - describe(name, fn)　　把相关测试组合在一起
  - test(name, fn)　　测试方法
  - expect(value)　　断言
- 匹配器
  - toBe(value)　　判断值是否相等
  - toEqual(obj)　　判断对象是否相等
  - toContain(value)　　判断数组或者字符串中是否包含
  - ……
- 快照
  - toMatchSnapshot()

Vue Test Utils 常用 API

- mount()
  - 创建一个包含被挂载和渲染的 Vue 组件的 Wrapper。
- Wrapper
  - vm　　wrapper 包裹的组件实例
  - props()　　返回 Vue 实例选项中的 props 对象
  - html()　　组件生成的 HTML 标签
  - find()　　通过选择器返回匹配到的组件中的 DOM 元素
  - trigger()　　触发 DOM 原生事件，自定义事件 wrapper.vm.$emit()
  - ……

Rollup 打包

Rollup

- Rollup 是一个模块打包器
- Rollup 支持 Tree-shaking
- 打包的结果比 Webpack 要小
- 开发框架/组件库的时候使用 Rollup 更合适

安装依赖

- Rollup
- rollup-plugin-terser
- rollup-plugin-vue@5.1.9
- vue-template-compiler

```
import { terser } from 'rollup-plugin-terser'
import vue from 'rollup-plugin-vue'
module.exports = [
  {
    input: 'index.js',
    output: [
      {
        file: 'dist/index.js',
        format: 'es'
      }
    ],
    plugins: [
      vue({
        css: true,
        compileTemplate: true
      }),
      terser()
    ]
  }
]
```

Rollup 打包

设置环境变量

清理

基于模板生成组件基本结构