打造 Vue.js 组件库

黄轶

黄 轶

Github：ustbhuangyi

- 北京科技大学毕业，计算机专业硕士
- 曾任职百度、滴滴，现担任Zoom前端架构师
- 慕课网明星讲师
- Vue.js 布道者，《Vue.js 技术揭秘》独立作者，《Vue.js 权威指南》主要作者
- 开源项目 better-scroll 作者，并主导滴滴开源项目 cube-ui，建立团队技术博客
- 对前端工程化，前后端性能优化有丰富的经验

目录
CONTENTS

背景

freeCodeCamp

背景

- 从业务角度看，业务成长到一定规模后，共性的地方需要复用。
- 从设计角度看，产品会遵循一定的设计规范，需要保证产品一致性。
- 从开发效率角度来看，需要快速响应业务开发。
- 从维护角度来看，需要统一管理代码，不希望到处复制粘贴。

组件设计

- 设计原则
- 模块依赖
- 开发规范

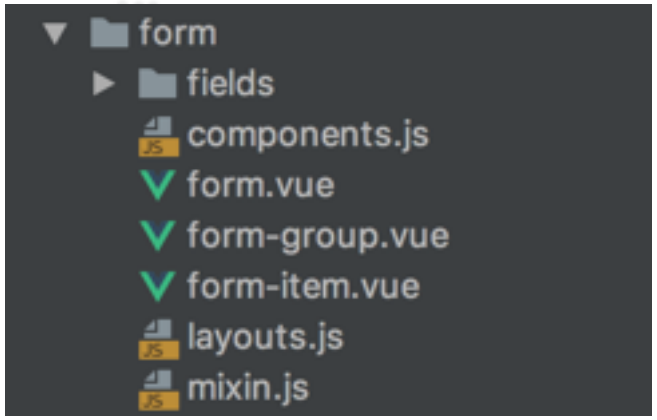
设计原则

就近管理　　高复用性　　分层设计　　灵活扩展

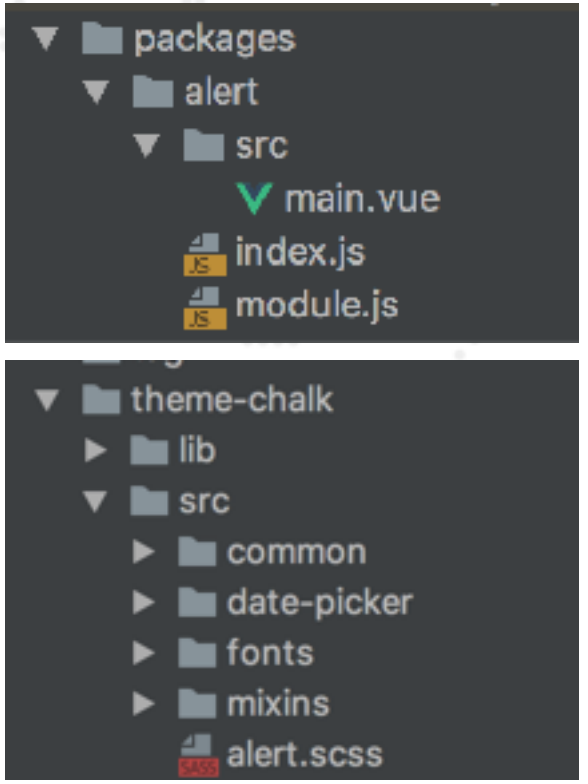
设计原则-就近管理

- 单文件开发
- 依赖的静态资源放在同级目录
- 相关联组件也放在同级目录

cube-ui



element-ui



设计原则-高复用性

- 页面级别的复用（基础组件）
- 项目级别的复用 —私有组件库（业务组件）
- 公司级别的复用 —开源组件库（element-ui、cube-ui）

设计原则-分层设计

通过分层设计的思想设计复杂组件

- 城市选择器组件、时间选择器组件
- 联动选择器 CascadePicker 实现
- 基础选择器 Picker 实现
- 弹层类组件 popup
- 基础滚动 better-scroll

设计原则-灵活扩展

组件设计要尽量灵活可扩展，除了提供丰富的 Props，还可以利用 slot 插槽完成用户个性化定制需求

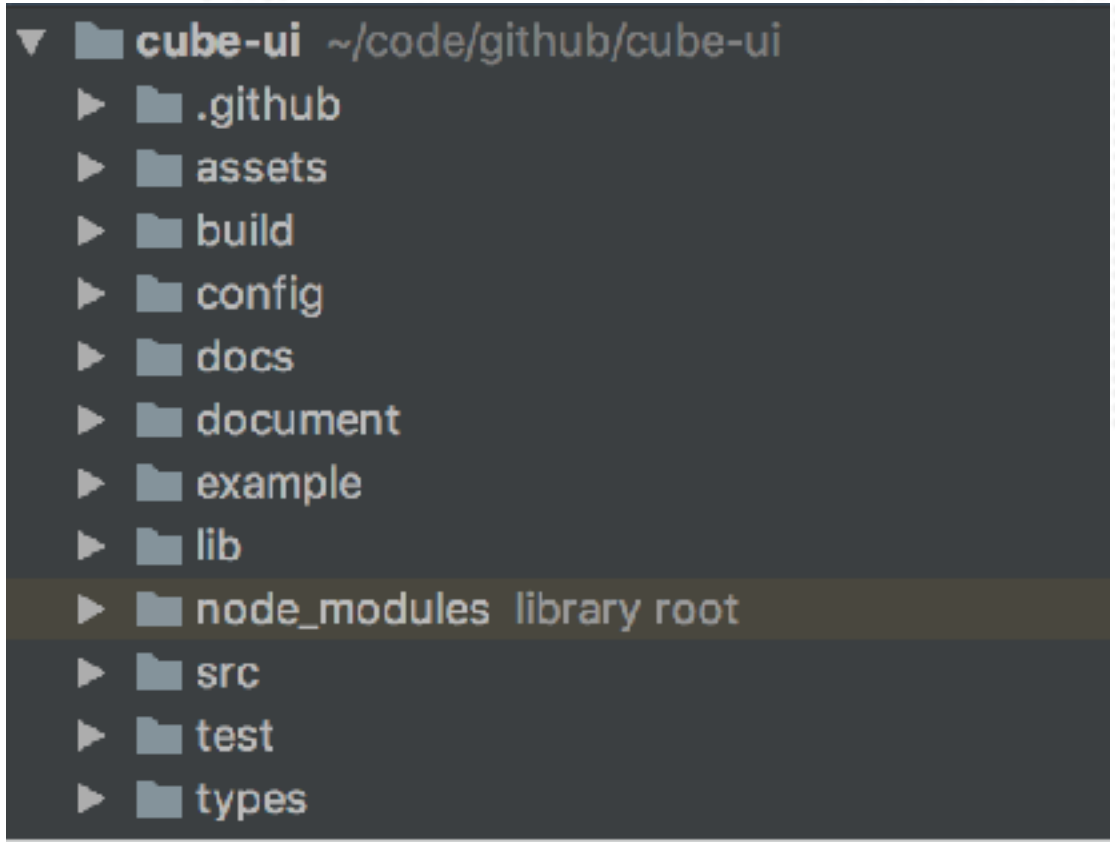
```
<cube-scroll
  ref="contentScroll"
  :data="content"
  :options="options"
  @pulling-down="onPullingDown"
  @pulling-up="onPullingUp">
  <ul class="imgs-wrapper">
    <li v-for="(item, index) in content" :key="index" class="imgs-item">
      <img :src="item.url" @load="onImgLoad">
    </li>
  </ul>
  <template slot="pulldown" slot-scope="props">
    <div v-if="props.pullDownRefresh"
      class="cube-pulldown-wrapper"
      :style="props.pullDownStyle">
      <div v-if="props.beforePullDown"
        class="before-trigger"
        :style="{paddingTop: props.bubbleY + 'px'}">
        <span :class="{rotate: props.bubbleY > 0}">↓</span>
      </div>
      <div class="after-trigger" v-else>
        <div v-show="props.isPullingDown" class="loading">
          <cube-loading></cube-loading>
        </div>
        <transition name="success">
          <div v-show="!props.isPullingDown" class="text-wrapper"><span
refresh-text">今日头条推荐引擎有x条更新</span></div>
        </transition>
      </div>
    </div>
  </template>
</cube-scroll>
```

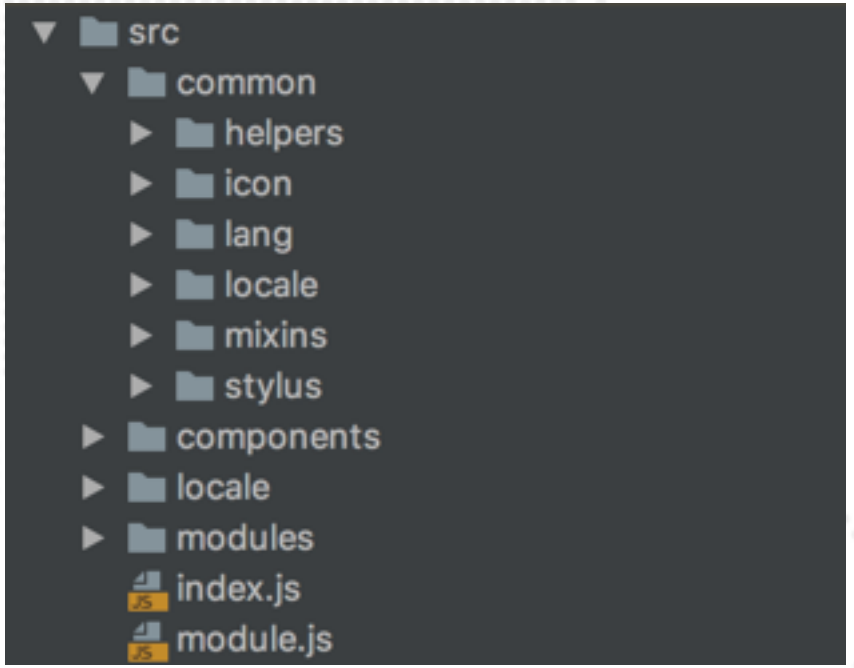
开发规范

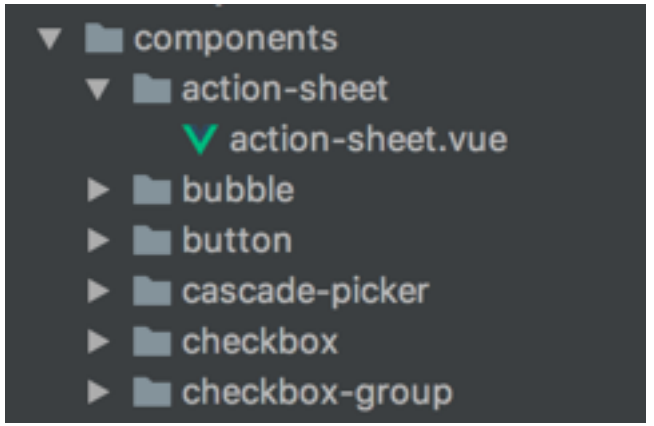
组件库需要设计统一的开发规范

开发规范-目录设计

cube-ui 目录结构







开发规范-代码编写

HTML 部分

- 使用 .vue 单文件开发组件，template 模板
- 尽量使用语义化标签

主要按钮

```
<div class="button">主要按钮</div>
```

```
<button class="button">主要按钮</button>
```

开发规范-代码编写

HTML 部分

- 遇到相同的结构考虑抽象出组件

开发规范-代码编写

HTML 部分

```
<div class="cube-popup" :style="{'z-index': zIndex}" :class="rootClass" v-show="isVisible">
  <div class="cube-popup-mask" @touchmove.prevent @click="maskClick">
    <slot name="mask"></slot>
  </div>
  <div class="cube-popup-container" @touchmove.prevent :class="containerClass">
    <div class="cube-popup-content" v-if="$slots.default">
      <slot></slot>
    </div>
    <div class="cube-popup-content" v-else v-html="content">
    </div>
  </div>
</div>
```

Popup

开发规范-代码编写

HTML 部分

```
<transition name="cube-picker-fade">
  <!-- Transition animation need use with v-show in the same template. -->
  <cube-popup
    type="picker"
    :mask="true"
    :center="false"
    :z-index="zIndex"
    v-show="isVisible"
    @touchmove.prevent
    @mask-click="maskClick">
    <transition name="cube-picker-move">
      <div class="cube-picker-panel cube-safe-area-pb" v-show="isVisible"
        <div class="cube-picker-choose border-bottom-1px"...>

        <div class="cube-picker-content"...>

        <div class="cube-picker-footer"></div>
      </div>
    </transition>
  </cube-popup>
</transition>
```

Picker

```
<transition name="cube-action-sheet-fade">
  <cube-popup
    type="action-sheet"
    :class="{'cube-action-sheet_picker': pickerStyle}"
    :center="false"
    :mask="true"
    :z-index="zIndex"
    v-show="isVisible"
    @mask-click="maskClick">
    <transition name="cube-action-sheet-move">
      <div class="cube-action-sheet-panel cube-safe-area-pb"
        <h1 class="cube-action-sheet-title border-bottom-1px"
title">{{title}}</h1>
        <div class="cube-action-sheet-content"...>
        <div class="cube-action-sheet-space"></div>
        <div class="cube-action-sheet-cancel"
@click="cancel"><span>{{_cancelTxt}}</span></div>
      </div>
    </transition>
```

ActionSheet

freeCodeCamp

开发规范-代码编写

CSS 部分

- CSS 预处理器（stylus、less、sass）、autoprefixer
- 定义全局变量：颜色、字体

```
$fontsize-large = 16px
$fontsize-medium = 14px
$fontsize-small = 12px
```

开发规范-代码编写

CSS 部分

- 定义全局 Mixin 函数

```
bg-image($url, $ext = ".png")
  background-image: url($url + "@2x" + $ext)
  @media (min-resolution: 3dppx)
    background-image: url($url + "@3x" + $ext)
```

- 组件样式使用 scope 或者 BEM 命名规范

```
<style lang="stylus" scoped>
```

```
<div class="dialog">
  <div class="el-dialog__wrapper">
    // dialog content
  </div>
</div>
```

开发规范-代码编写

JS 部分

- ES2015、eslint
- 避免魔术字符串和魔术数字、用常量替代

```
const COMPONENT_NAME = 'cube-dialog'
const EVENT_CONFIRM = 'confirm'
const EVENT_CANCEL = 'cancel'
const EVENT_CLOSE = 'close'
```

```
close(e) {
  this.hide()
  this.$emit(EVENT_CLOSE, e)
}
```

开发规范-代码编写

JS 部分

- 组件遇到相同的逻辑可以抽象出 Mixin

```
export default {
  props: {
    zIndex: {
      type: Number,
      default: 100
    },
    maskClosable: {
      type: Boolean,
      default: false
    }
  }
}
```

PopopMixin

```
const COMPONENT_NAME = 'cube-dialog'

export default {
  name: COMPONENT_NAME,
  mixins: [visibilityMixin, popupMixin, localeMixin],
  props: {
    type: {
      type: String,
      default: 'alert'
    },
```

Dialog

```
const COMPONENT_NAME = 'cube-picker'

export default {
  name: COMPONENT_NAME,
  mixins: [visibilityMixin, popupMixin,
  props: {
```

Picker

开发规范-代码编写

JS 部分

- 父组件往子孙组件注入依赖可以考虑用 provide/inject

```
const COMPONENT_NAME = 'cube-scroll-nav'

export default {
  name: COMPONENT_NAME,
  provide() {
    return {
      scrollNav: this
    }
  },
```

ScrollNav

ScrollNav

```
const COMPONENT_NAME = 'cube-scroll-nav-bar'

export default {
  name: COMPONENT_NAME,
  inject: {
    scrollNav: {
      default: null
    }
  },
```

ScrollNavBar

开发规范-代码编写

JS 部分

- 尽量避免写 hack 的代码，

  如需要，一定要写明注释

```
created() {
  this._dataWatchers = []
  const needRefreshProps = ['data', 'loop', 'autoPlay', 'options.eventPassthrough',
threshold', 'speed', 'allowVertical']
  needRefreshProps.forEach((key) => {
    this._dataWatchers.push(this.$watch(key, () => {
      // To fix the render bug when add items since loop.
      if (key === 'data') {
        this._destroy()
      }

      /* istanbul ignore next */
      this.$nextTick(() => {
        this.refresh()
      })
    }))
  })
},
```

Slide

开发规范-文档和测试

每个组件都要有详细的使用文档、以及示例代码

文档和测试

需要提供组件库的安装、快速上手等文档

需要有完整的单元测试、测试覆盖率达到 90% 以上

模块依赖

组件库可以依赖一些核心模块作为辅助。

模块依赖- vue-create-api

- 一个能够让 Vue 组件通过 API 方式调用的插件。
- 动态把组件挂载到 body 下。
- 支持传入响应式 props、events、插槽、单例和多例模式。
- 在普通 JS 文件中也能调用组件。

模块依赖- vue-create-api

支持把任意组件变成 API 式的调用

```
import Vue from 'vue'
import CreateAPI from 'vue-create-api'

Vue.use(CreateAPI)

// 也可以传递一个配置项

Vue.use(CreateAPI, {
  componentPrefix: 'cube-',
  apiPrefix: '$create-'
})

// 之后会在 Vue 构造器下添加一个 createAPI 方法
```

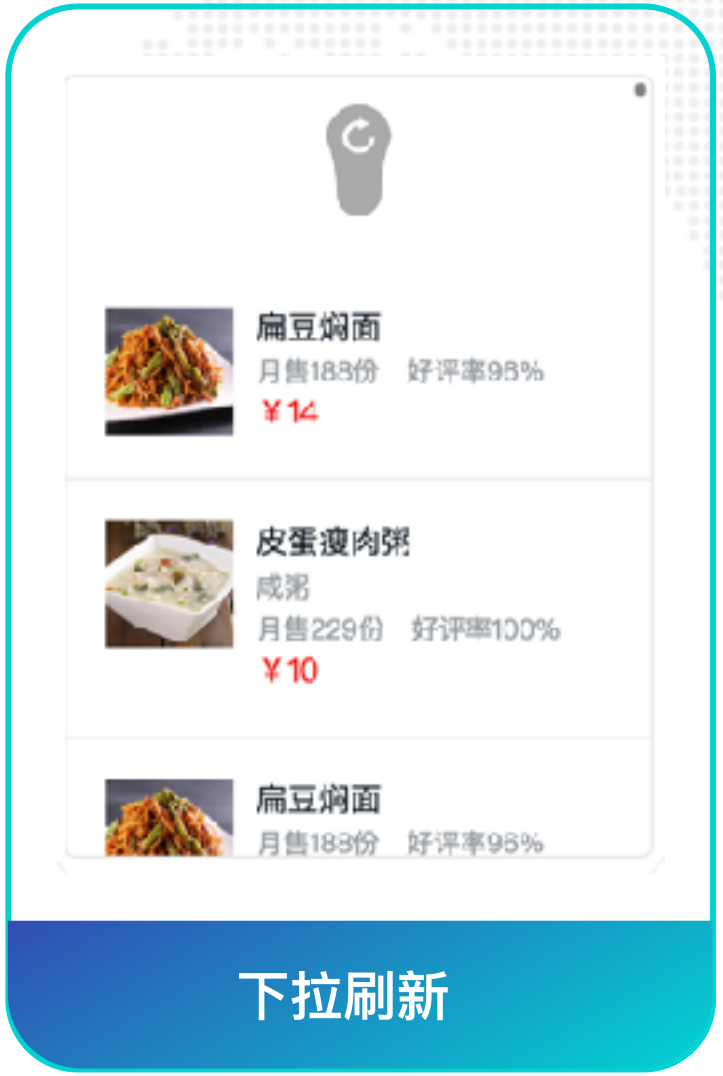
```
import Dialog from './components/dialog.vue'

// 调用 createAPI 生成对应 API，并挂载到 Vue.prototype 和 Diclog 对象上

Vue.createAPI(Dialog, true)

// 之后便可以在普通的 js 文件中使用，但是 $props 不具有响应式

Dialog.$create({
  $props: {
    title: 'Hello',
    content: 'I am from pure JS'
  }
}).show()

// 在 vue 组件中可以通过 this 调用

this.$createDialog({
  $props: {
    title: 'Hello',
    content: 'I am from a vue component'
  },
}).show()
```

模块依赖- better-scroll

- 重点解决移动端（支持 PC 端）各种滚动场景需求的插件。
- 原生 JS 实现，可配合任意 MVVM 框架使用。
- 支持丰富的配置。

模块依赖- better-scroll

普通滚动



下拉刷新

上拉加载

模块依赖- better-scroll

Picker

索引列表

滚动导航

模块依赖- better-scroll

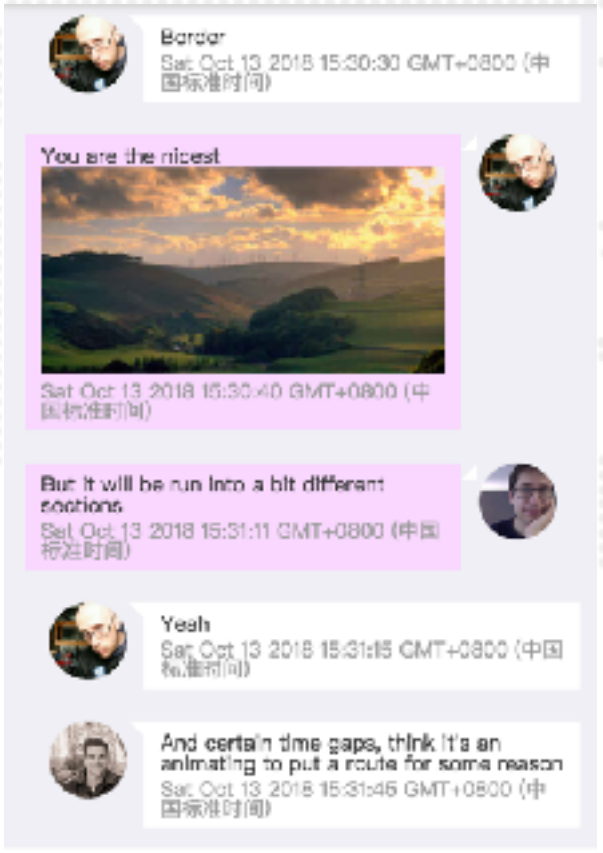
TabBar

开屏引导

轮播图

模块依赖- better-scroll

图片预览



无限滚动

模块依赖- poper.js

- 重点解决 PC 端弹出层类需求的插件。
- 它是一个定位引擎，通过动态计算元素的位置，将其定位在给定的参考元素附近。
- 原生 JS 实现，可配合任意 MVVM 框架使用。

模块依赖- poper.js

选择器

文字提示

（右边空间不够）

模块依赖- poper.js

日期选择

弹出框

（下边空间不够）

工程化

- npm scripts
- 单元测试
- 打包部署
- 生态建设

npm scripts

1

介绍

2

常见场景

npm scripts- 介绍

npm 允许在package.json文件里面，使用scripts字段定义脚本命令

```
"scripts": {
  "build": "node build/build.js"
},
```

命令行下使用npm run命令，就可以执行这段脚本。

```
npm run build
```

等同于执行

```
node build/build.js
```

npm scripts- 常见场景

cube-ui 的 npm scripts

```
"scripts": {
  "build": "node build/build.js",
  "dev": "node build/dev-server.js",
  "doc-dev": "npm run dev & node build/document/dev-server.js",
  "doc-build": "node build/document/build.js",
  "demo-build": "node build/example/build.js",
  "doc-demo-build": "npm run doc-build && npm run demo-build",
  "release": "bash ./build/release/publish.sh",
  "release-docs": "bash ./build/release/docs.sh",
  "lint": "eslint --ext .js,.vue src test/unit/specs test/e2e/specs",
  "unit": "cross-env BABEL_ENV=test karma start test/unit/karma.conf.js --single-run",
  "codecov": "codecov",
  "test": "npm run unit && npm run codecov",
  "cm": "git-cz"
},
```

npm scripts- 常见场景

element-ui 的 npm scripts

```
"scripts": {
  "bootstrap": "yarn || npm i",
  "build:file": "node build/bin/iconInit.js & node build/bin/build-entry.js & node
build/bin/i18n.js & node build/bin/version.js",
  "build:theme": "node build/bin/gen-cssfile && gulp build --gulpfile
packages/theme-chalk/gulpfile.js && cp-cli packages/theme-chalk/lib lib/theme-chalk",
  "build:utils": "cross-env BABEL_ENV=utils babel src --out-dir lib --ignore src/index.js",
  "build:umd": "node build/bin/build-locale.js",
  "clean": "rimraf lib && rimraf packages/*/lib && rimraf test/**/coverage",
  "deploy": "npm run deploy:build && gh-pages -d examples/element-ui --remote eleme &&
rimraf examples/element-ui",
  "deploy:build": "npm run build:file && cross-env NODE_ENV=production webpack --config
build/webpack.demo.js && echo element.eleme.io>>examples/element-ui/CNAME",
  "dev": "npm run bootstrap && npm run build:file && cross-env NODE_ENV=development
webpack-dev-server --config build/webpack.demo.js & node build/bin/template.js",
  "dev:play": "npm run build:file && cross-env NODE_ENV=development PLAY_ENV=true
webpack-dev-server --config build/webpack.demo.js",
  "dist": "npm run clean && npm run build:file && npm run lint && webpack --config
build/webpack.conf.js && webpack --config build/webpack.common.js && webpack --config
build/webpack.component.js && npm run build:utils && npm run build:umd && npm run build:theme"
  "i18n": "node build/bin/i18n.js",
  "lint": "eslint src/**/* test/**/* packages/**/* build/**/* --quiet",
  "pub": "npm run bootstrap && sh build/git-release.sh && sh build/release.sh && node
build/bin/gen-indices.js && sh build/deploy-faas.sh",
  "test": "npm run lint && npm run build:theme && cross-env CI_ENV=/dev/ karma start
test/unit/karma.conf.js --single-run",
  "test:watch": "npm run build:theme && karma start test/unit/karma.conf.js"
},
```

单元测试

1

介绍

2

编写测试用例

单元测试-介绍

用于测试组件模块的行为是否能达到预期结果的代码

- 保证组件库库在后续的开发维护的稳定性
- 降低人工测试的成本

单元测试-介绍

- 测试工具 karma
- 测试框架 mocha
- 断言工具 chai
- 测试覆盖率

单元测试-编写测试脚本

```
describe('Button.vue', () => {
  let vm
  afterEach(() => {
    if (vm) {
      vm.$parent.destroy()
      vm = null
    }
  })
  it('use', () => {
    Vue.use(Button)
    expect(Vue.component(Button.name))
      .to.be.a('function')
  })
  it('should render correct contents', () => {
    vm = instantiateComponent(Vue, Button, {
      props: {
        icon: 'mfic-back'
      }
    }, (createElement) => {
      return createElement('span', 'btn content')
    })
    expect(vm.$el.querySelector('i').className)
      .to.equal('mfic-back')
    expect(vm.$el.querySelector('span').textContent)
      .to.equal('btn content')
  })
```

describe 块称为“测试套件”（test suite），表示一组相关的测试

it 块称为"测试用例"（test case），表示一个单独的测试，是测试的最小单位

expect 表示断言，判断源码的实际执行结果与预期结果是否一致，如果不一致就抛出一个错误。

打包部署

入口 JS 编写

Webpack

构建配置

部署脚本

打包部署-入口 JS 编写

以 cube-ui 为例

```
// 拿到所有的组件
const components = [...]

function install(Vue) {
  if (install.installed) {
    return
  }
  install.installed = true
  components.forEach((Component) => {
    // ignore radio
    if (Component === Radio) {
      return
    }
    Component.install(Vue)
  })
}
```

```
const Cube = {
  /* eslint-disable no-undef */
  version: __VERSION__,
  install,
  BScroll: BetterScroll,
  createAPI
}

components.forEach((Component) => {
  const name = processComponentName(Component, {
    firstUpperCase: true
  })
  Cube[name] = Component
})

if (typeof window !== 'undefined' && window.Vue) {
  window.Vue.use(install)
}

export default Cube
```

打包部署-入口 JS 编写

以 TimePicker 组件为例

```
import Picker from '../../components/picker/picker.vue'
import TimePicker from '../../components/time-picker/time-picker.vue'
import addTimePicker from './api'
import addPicker from '../picker/api'

TimePicker.install = function (Vue) {
  Vue.component(Picker.name, Picker)
  Vue.component(TimePicker.name, TimePicker)
  addPicker(Vue, Picker)
  addTimePicker(Vue, TimePicker)
}

TimePicker.Picker = Picker

export default TimePicker
```

打包部署-webpack 构建配置

以 cube-ui 组件为例

```
var webpackConfig = merge(baseWebpackConfig, {
  entry: {
    cube: './src/index.js'
  },
  module: {
    rules: utils.styleLoaders({
      sourceMap: config.build.productionSourceMap,
      extract: true
    })
  },
  devtool: config.build.productionSourceMap ? '#source-map' : false,
  output: {
    path: config.build.assetsRoot,
    filename: utils.assetsPath('index.js'),
    library: 'cube',
    libraryTarget: 'umd'
  },
  plugins: [
    // extract css into its own file
    new ExtractTextPlugin(utils.assetsPath('style.css'))
  ]
})
```

打包部署-编写构建脚本

cube-ui 的构建脚本

```
#!/bin/bash

# git pull
git pull origin master
# npm install
rm package-lock.json
npm install --registry=https://registry.npmjs.org
# build
npm run build
# ADD commit
git add -A
git commit -m 'build: package'
git push origin master
# replace src/ __VERSION__
node ./build/release/replace-version.js
# publish
npm publish
# checkout src/index.js
git checkout src/index.js
git checkout dev
git rebase master
git push origin dev
git checkout master
```

生态建设

脚手架

上手教程

按需引入

后编译

Q&A

生态建设-脚手架

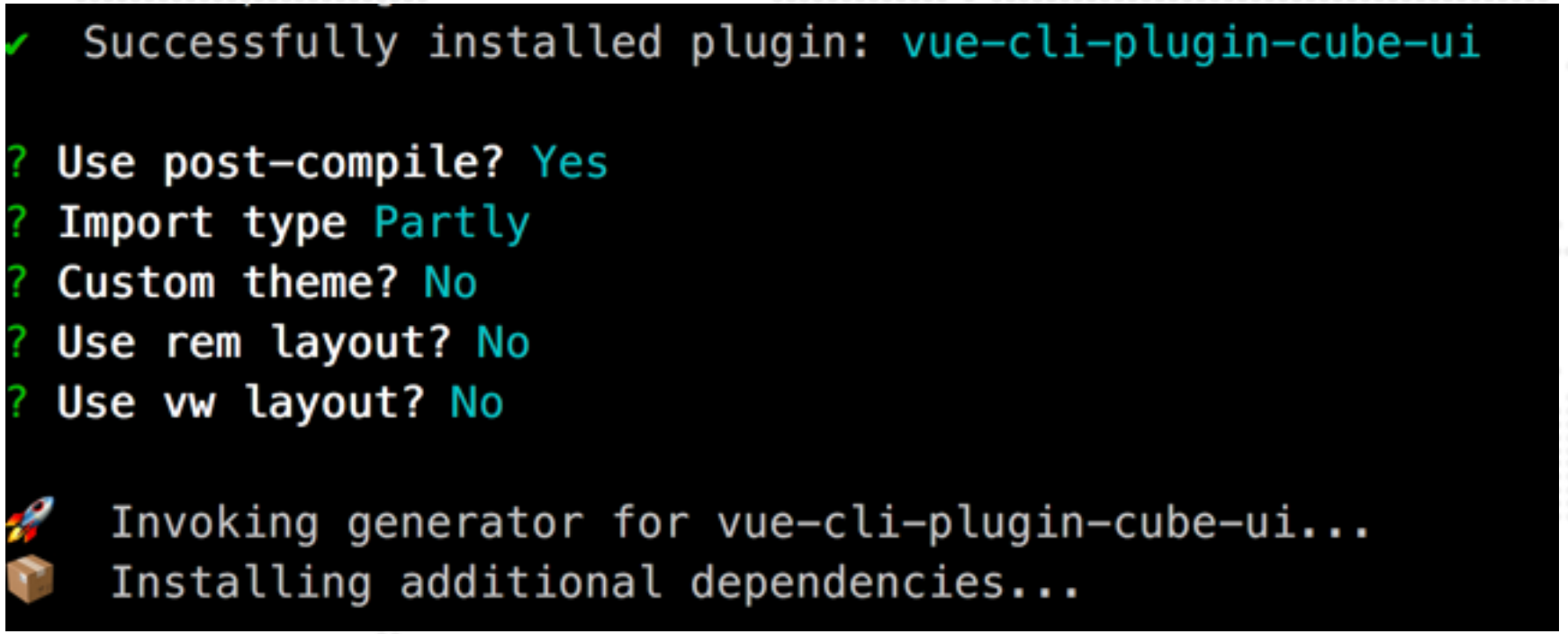
脚手架可以帮我们快速生成项目的初始化代码。

Vue 官方提供了 vue-cli 脚手架帮助我们快速生成 Vue 项目

Vue-cli 3.0 提供了插件化机制允许我们给初始化项目注入代码和配置

- **vue-cli-plugin-element**
- **vue-cli-plugin-cube-ui**

生态建设-脚手架



- 添加组件引入代码
- 新增 vue.config.js 扩展 webpack 配置
- 新增依赖插件并下载安装

生态建设-上手教程

帮助用户快速学习和了解组件库的使用

- cube-application-guide

生态建设-按需引入

只引入需要的组件，以达到减小项目体积的目的

```
import Button from 'cube-ui/lib/button'
import 'cube-ui/lib/button/style.css'
```

```
import { Button } from 'cube-
```

右边在左边显示完出现

借助 webpack-transform-modules-plugin

```
"transformModules": {
  "cube-ui": {
    "transform": "cube-ui/lib/${member}",
    "kebabCase": true,
    "style": {
      "ignore": ["create-api", "better-scr
    }
  }
}
```

下面要在上面展示完显示

生态建设-后编译

编译代码冗余?

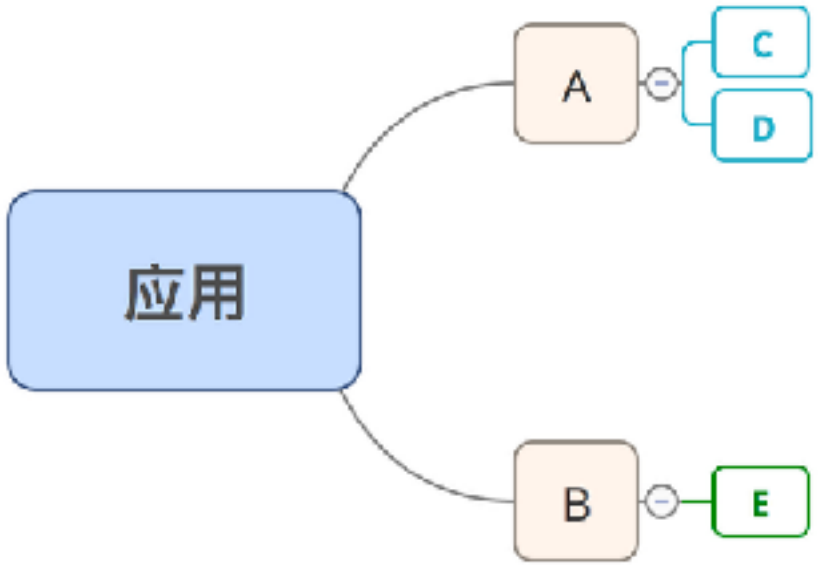
依赖包本身不编译，它的编译交给应用来做。

通过修改 webpack 配置 rules 中的 include

```
module: {
  rules: [
    {loader: 'eslint-loader'...},
    {loader: 'vue-loader'...},
    {
      test: /\.js$/,
      loader: 'babel-loader',
      include: allSource.concat(resolve('node_modules/lodash-es'))
    },
```

生态建设-后编译

后编译依赖嵌套?



webpack-post-compile-plugin

需要后编译的依赖包在 package.json 中声明

```
"postCompile": true,
```

生态建设-后编译

- 一份编译代码
- 一份 ployfill
- NPM包无需编译发布
- 主题定制
- rem布局

生态建设-Q&A

整理常见的问题问答

FCC
freeCodeCamp

Q&A

慕课网 imooc.com

ustbhuangyi

Vue.js 高仿开发
饿了么APP

完整流程

Vue 2.0实战高级-
开发移动端音乐
Web APP

复杂应用

Vue.js 源码
全方位深入解析

高级进阶

freeCodeCamp

黄轶