一、创建项目，（ 首先mkdir mydemo

进入根目录vue create . (同vue create mydemo注意创建项目名全小写，不能驼峰命名) ）

二、

我们需要一个目录存放组件，一个目录存放示例，按照以下方式对目录进行改造。

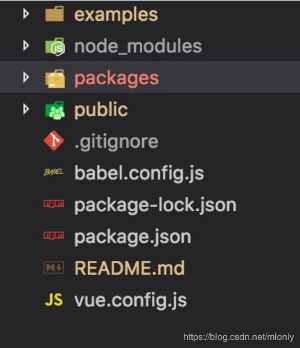
.

…

|-- examples // 原 src 目录，改成 examples 用作示例展示

|-- packages // 新增 packages 用于编写存放组件

…



四、配置项目以支持新的目录结构

我们通过上一步的目录改造后，会遇到两个问题。

src目录更名为examples，导致项目无法运行

新增packages目录，该目录未加入webpack编译

注：cli3 提供一个可选的 vue.config.js 配置文件。如果这个文件存在则他会被自动加载，所有的对项目和webpack的配置，都在这个文件中。

1、重新配置入口，修改配置中的 pages 选项

新版 Vue CLI 支持使用 vue.config.js 中的 pages 选项构建一个多页面的应用。

这里使用 pages 修改入口到 examples

2、支持对 packages 目录的处理，修改配置中的 chainWebpack 选项

packages 是我们新增的一个目录，默认是不被 webpack 处理的，所以需要添加配置对该目录的支持。

chainWebpack 是一个函数，会接收一个基于 webpack-chain 的 ChainableConfig 实例。允许对内部的 webpack 配置进行更细粒度的修改。

module.exports = {

// 修改 src 为 examples

pages: {

index: {

entry: 'examples/main.js',

template: 'public/index.html',

filename: 'index.html'

}

},

// 扩展 webpack 配置，使 packages 加入编译

chainWebpack: config => {

config.module

.rule('js')

.include

.add('/packages')

.end()

.use('babel')

.loader('babel-loader')

.tap(options => {

// 修改它的选项...

return options

})

}

// 是否提取css到单独的文件中，库构建时为false以免将css单独提取到一个文件里。

,css{

extract:false

}

}

五、编写组件

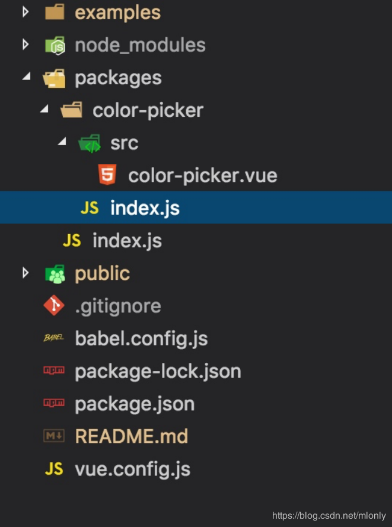
以上我们已配置好对新目录架构的支持，接下来我们尝试编写组件。

1、创建一个新组件

在 packages 目录下，所有的单个组件都以文件夹的形式存储，所有这里创建一个目录 color-picker/

在 color-picker/ 目录下创建 src/ 目录存储组件源码

在 /color-picker 目录下创建 index.js 文件对外提供对组件的引用。



修改 /packages/color-picker/index.js文件，对外提供引用。

# /packages/color-picker/index.js

// 导入组件，组件必须声明 name

import colorPicker from './src/color-picker.vue'

// 为组件提供 install 安装方法，供按需引入

colorPicker.install = function (Vue) {

Vue.component(colorPicker.name, colorPicker)

}

// 默认导出组件

export default colorPicker

2、

整合所有的组件，对外导出，即一个完整的组件库

修改 /packages/index.js 文件，对整个组件库进行导出。

// 导入颜色选择器组件

import colorPicker from ‘./color-picker’

// 存储组件列表

const components = [

colorPicker

]

// 定义 install 方法，接收 Vue 作为参数。如果使用 use 注册插件，则所有的组件都将被注册

const install = function (Vue) {

// 判断是否安装

if (install.installed) return

// 遍历注册全局组件

components.map(component => Vue.component(component.name, component))

}

// 判断是否是直接引入文件

if (typeof window !== ‘undefined’ && window.Vue) {

install(window.Vue)

}

export default {

// 导出的对象必须具有 install，才能被 Vue.use() 方法安装

install,

// 以下是具体的组件列表

colorPicker

}





七、发布到 npm，方便直接在项目中引用

到此为止我们一个完整的组件库已经开发完成了，接下来就是发布到 npm 以供后期使用。

1、package.json 中新增一条编译为库的命令

在库模式中，Vue是外置的，这意味着即使在代码中引入了 Vue，打包后的文件也是不包含Vue的。

Vue Cli3 构建目标：库

以下我们在 scripts 中新增一条命令 npm run lib

–target: 构建目标，默认为应用模式。这里修改为 lib 启用库模式。

–dest : 输出目录，默认 dist。这里我们改成 lib

[entry]: 最后一个参数为入口文件，默认为 src/App.vue。这里我们指定编译 packages/ 组件库目录。

“scripts”: {

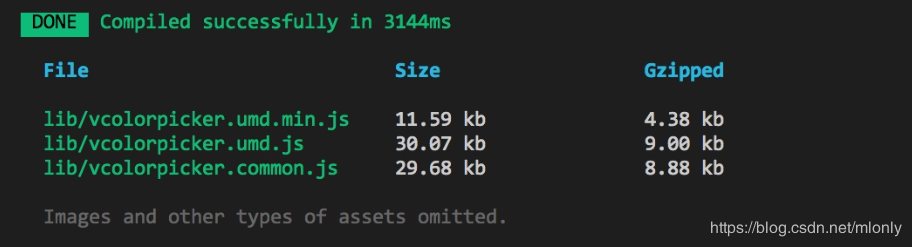
// …

“lib”: “vue-cli-service build --target lib --name vcolorpicker --dest lib packages/index.js”

}

执行编译库命令

$ npm run lib



2、配置 package.json 文件中发布到 npm 的字段

name: 包名，该名字是唯一的。可在 npm 官网搜索名字，如果存在则需换个名字。

version: 版本号，每次发布至 npm 需要修改版本号，不能和历史版本号相同。

description: 描述。

main: 入口文件，该字段需指向我们最终编译后的包文件。

keyword：关键字，以空格分离希望用户最终搜索的词。

author：作者

private：是否私有，需要修改为 false 才能发布到 npm

license： 开源协议

以下为参考设置

{

“name”: “vcolorpicker”,

“version”: “0.1.5”, //每次更新都需要修改版本号

“description”: “基于 Vue 的颜色选择器”,

“main”: “lib/vcolorpicker.umd.min.js”,

“keyword”: “vcolorpicker colorpicker color-picker”,

“private”: false

}

3、添加 .npmignore 文件，设置忽略发布文件

我们发布到 npm 中，只有编译后的 lib 目录、package.json、README.md才是需要被发布的。所以我们需要设置忽略目录和文件。

在根目录下添加.npmignore，具体内容为

#忽略目录

examples/

packages/

public/

#忽略指定文件

vue.config.js

babel.config.js

\*.map

4、登录到 npm

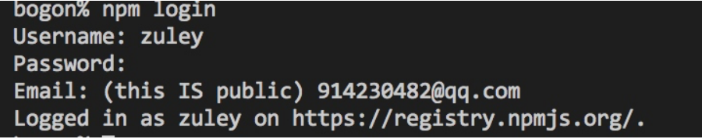
首先需要到 npm 上注册一个账号，注册过程略。

如果配置了淘宝镜像，先设置回npm镜像：

$ npm config set registry http://registry.npmjs.org

然后在终端执行登录命令，输入用户名、密码、邮箱即可登录。

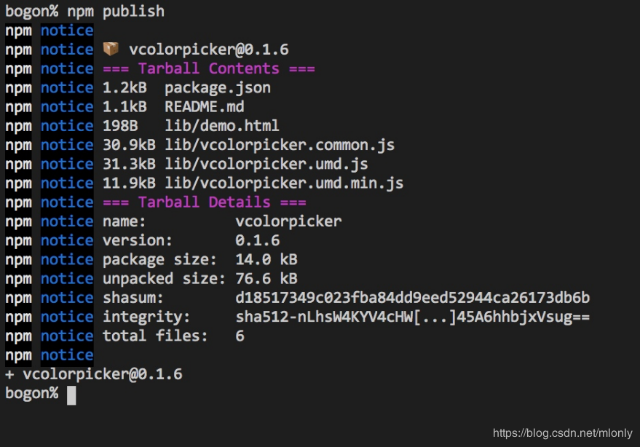
npm login



5、发布到 npm

执行发布命令，发布组件到 npm

$ npm publish



6、npm install sujinfengdemo –S //安装

npm uninstall sujinfengdemo –S //卸载

# 在 main.js 引入并注册

import vcolorpicker from 'vcolorpicker'

Vue.use(vcolorpicker)