

课程目标

业务组件库搭建

市面上目前已有各种各样的 UI 组件库，比如 `Element` 和 `iView`，他们都是一些基础组件库，但是我们面临的情况是需求越来越复杂，不同业务组件在很多系统，或者某个系统中多次使用，所以一般我们多系统业务类似的时候，单一的单文件组件就不太适用我们，不再满足我们的需求，这个时候就有必要开发一套属于自己团队的业务组件库了。基础组件库就不带大家开发，目前有现成的基础组件库，像 `element` 和 `iview` 等等，常用的 ui 组件库如下：

移动端 UI 组件库： Mint UI Vant UI cube-ui

PC 端的 UI 组件库： iView Element UI Ant Design Vue

搭建组件库的步骤：

- 搭建开发组件库环境
- 配置项目支持目录结构打包编译
- 编写组件
- 配置使用库模式打包编译
- 发布到 `npm`
- 下载使用

1、搭建开发组件库环境

- 创建项目

通过vue-cli搭建组件开发环境，在指定目录中使用命令创建一个默认的项目，或者根据自己需要自己选择。 `vue create gupao2`

- 调整目录

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  
...  
.
```

2、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。

1. `src` 目录更名为 `examples`，导致项目无法运行
2. 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 and `webpack` 的配置，都在这个文件中。

(1) 重新配置入口，修改配置中的 `page` 选项：

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`

```
module.exports = {  
  // 修改 src 目录 为 examples 目录  
  pages: {  
    index: {  
      entry: 'examples/main.js',  
      template: 'public/index.html',  
      filename: 'index.html'  
    }  
  }  
}
```

(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。

webpack-chain: 是什么? 经过链式的方式修改webpack的配置

```
// 若是是非webpack-chain的话
module:{
  rules:[
    {
      test: /\.vue/,
      use:[
        {
          loader: "vue-loader",
          options: {}
        }
      ]
    }
  ]
}
// 一个例子
config.module
  .rule('vue')
  .test(/\.vue$/)
  .use('vue-loader')
  .loader('vue-loader')
  .tap(options => {
    return options
  })
  .end()
```

```
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
        })
  }
}
```

3、编写组件

- (1)、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 `image-preview/`
- (2)、在 `image-preview/` 目录下创建 `src/` 目录存储组件源码
- (3)、在 `/image-preview` 目录下创建 `index.js` 文件对外提供对组件的引用。

```
// 导入组件，组件必须声明 name
import ImagePreview from './src/ImagePreview.vue'

// 为组件提供 install 安装方法，供按需引入
ImagePreview.install = function (Vue) {
  Vue.component(ImagePreview.name, ImagePreview)
}
// 默认导出组件
export default ImagePreview
```


(4)、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

```
// 导入图片预览组件
import ImagePreview from './image-preview'

// 存储组件列表
const components = [
  ImagePreview
]

// 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
  ImagePreview
}
```

(5)、编写示例(记着基于element-ui的业务组件需要先引入element)

```
import Vue from 'vue'
import App from './App.vue'
// 导入组件库
import Gupao1 from '../packages/index'
import ElementUI from 'element-ui';
import 'element-ui/lib/theme-chalk/index.css';
Vue.use(ElementUI);
// 注册组件库
Vue.use(Gupao1)
Vue.config.productionTip = false

new Vue({
  render: h => h(App),
}).$mount('#app')
```

(6)、在示例中使用组件库中的组件

```
<template>
  <div id="app">
    <image-preview :list="list" ref="imgs"></image-preview>
    
  </div>
</template>
<script>
export default {
  name: 'App',
  methods:{
    add() {
      this.$refs.imgs.open(0)
    },
  },
  data() {
    return {
      list2:
        ['https://ss0.bdstatic.com/70cFuHSh_Q1YnxGkpoWK1HF6hhy/it/u=3244460882,2086599540&fm=26&gp=0.jpg','https://s3.bdstatic.com/70cFv8Sh_Q1YnxGkpoWK1HF6hhy/it/u=3824886304,665215047&fm=26&gp=0.jpg'],
    }
  }
}
</script>
```


4、配置使用库模式打包编译

`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。

以下我们在 `scripts` 中新增一条命令 `npm run lib`

```
"scripts": {  
  // ...  
  "lib": "vue-cli-service build --target lib --name gupao1 --dest lib packages/index.js"  
}
```

`--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。

执行编译库命令 `npm run lib`

lib\gupao1.umd.min.js	13.91 KiB	5.1
lib\gupao1.umd.js	47.06 KiB	11.
lib\gupao1.common.js	46.60 KiB	11.
lib\gupao1.css	3.54 KiB	

5、发布到 npm

(1)、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

参考设置：

```
"name": "gupao1",  
"version": "1.1.0",  
"private": false,  
"author": "一休",  
"description": "基于 Vue 的图片预览",  
"main": "lib/gupao1.umd.min.js",
```

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。和 `.gitignore` 的语法一样，具体需要提交什么文件，看各自的实际情况

```
# 忽略目录  
examples/  
packages/  
public/  
  
# 忽略指定文件  
vue.config.js  
babel.config.js
```

(2)、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 get registry 查看当前源
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`

登录成功后执行发布命令，发布组件到 [npm](#) `npm publish`

```
.....
npm notice 306B packages/ts-crud/index.js
npm notice 173B packages/ts-crud/src/ts-crud.vue
npm notice 4.3kB public/favicon.ico
npm notice 611B public/index.html
npm notice 566B vue.config.js
npm notice === Tarball Details ===
npm notice name: gupao1
npm notice version: 1.1.0
npm notice filename: gupao1-1.1.0.tgz
npm notice package size: 106.3 kB
npm notice unpacked size: 363.1 kB
npm notice shasum: 9be4a9678d85af4d64336445e56eb980e5304f56eb980e5304f
O0vhdA0xCHQ0Q==
npm notice integrity: sha512-hviJrwCfQXQy+[...]O0vhdA0xCHQ0Q==
npm notice total files: 29
npm notice
+ gupao1@1.1.0
```

发布成功后稍等几分钟，即可在 [npm](#) 官网搜索到。以下是刚提交的 gupao1

6、使用新发布的组件库

yarn add gupao1, 因为是基于element-ui的业务组件, 所以先要下载 yarn add element-ui

```
import Vue from 'vue'
import App from './App.vue'
import ElementUI from 'element-ui';
import 'element-ui/lib/theme-chalk/index.css';
import Gupao1 from 'gupao1'
Vue.use(ElementUI);
Vue.use(Gupao1)
Vue.config.productionTip = false

new Vue({
  render: h => h(App),
}).$mount('#app')
```

即可按照示例使用