

Vue-Cli的HelloWorld程序



软通大学
ISOFTSTONE UNIVERSITY

• 本节目标

- ◆ 理解SPA
- ◆ 理解MVVM架构
- ◆ 认识Vue-Cli项目标准目录结构
- ◆ 认识Vue-Cli项目启动过程





目录 CONTENTS

1 前言

MVVM架构 2

3 VueCli项目目录解析

VueCli项目启动过程 4

5 本章总结

01 前言



● 前言

前后端分离是目前WEB开发中的一个大的趋势，随着各种前后端框架的出现，加上REST编程规范慢慢深入人心，前端后端之间通过更加轻量级、简洁高效的JSON作为数据传输格式，使得前后端分工更加明晰，前端被赋予了更多的功能，从而能分担原来由后端完成的工作。

前后端分离的例子就是SPA (Single-page application)，所有用到的展现数据都是后端通过异步接口 (AJAX/JSONP) 的方式提供。后端只负责数据，前端围绕数据而展开的数据逻辑，视图效果都由前端项目独立完成

单页面应用：**single-page application**:

单页面应用指的就是只有一张Web页面的应用，是加载单个HTML页面并在用户与应用程序交互时动态更新该页面的web应用程序。

单页应用是一种网络应用程序或网站的模型，它通过动态重写当前页面来与用户交互，而非传统的从服务器重新加载整个新页面。这种方法避免了页面之间切换打断用户体验，使应用程序更像一个桌面应用程序。

特点：本地应用一般的速度和流畅、MVVM 开发模式，前后端各负其责，后端采用mvc结构，前端视图层采用mvvm、业务逻辑全部都在本地操作，数据都需要通过 ajax 同步提交、在 URL 中采用 # 号来作为当前视图的地址 (http://xxx.com/#/)，改变 # 号后的参数来载入不同的页面片段(页面并不会重载！)



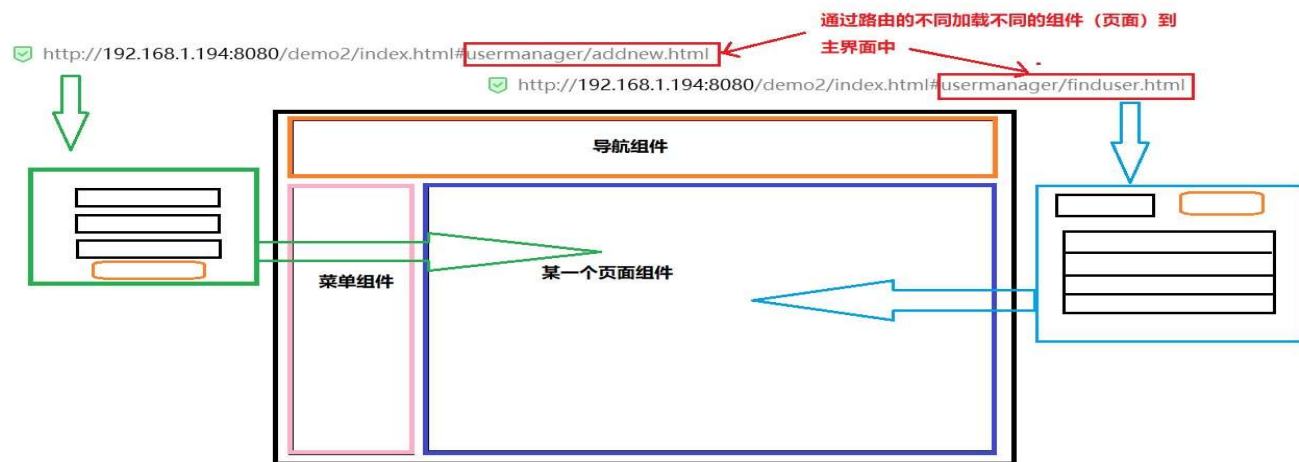
02 MVVM架构



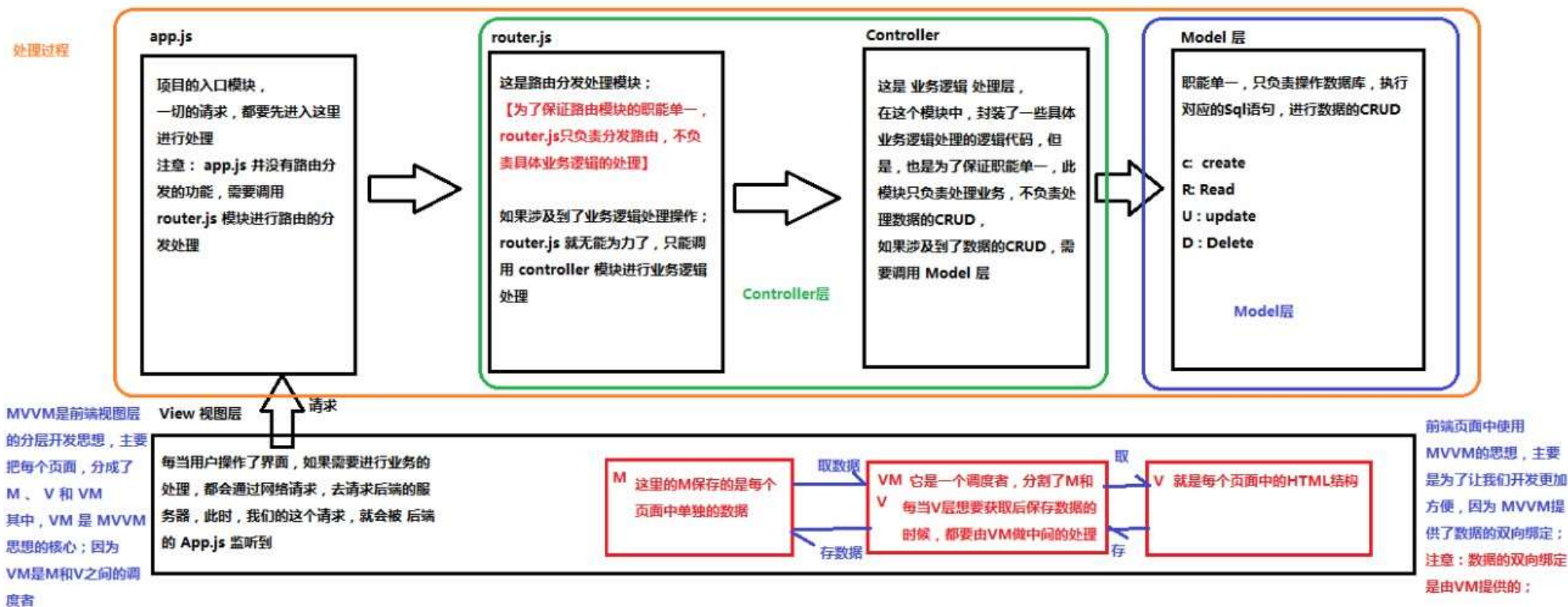
• MVVM架构-Vue工作模式

Vue的工作模式就是通过MVVM架构思想而设计实现的:

- 数据驱动: 数据改变驱动了视图的自动更新, 传统的做法是程序代码手动改变DOM来改变视图, vuejs只需要改变数据, 就会自动改变视图, 换言之, 用**vue, 就不能使用js操作dom**, 而是vue的数据改变, 自动更新视图的变化。
- 组件化: 把整个网页的拆分成一个个组件(区块), 每个区块作成一个组件。网页由多个组件拼接或者嵌套组成。而组件可以复用。所以在Vue.js中, 网页是可以看成多个组件组成



● MVVM架构



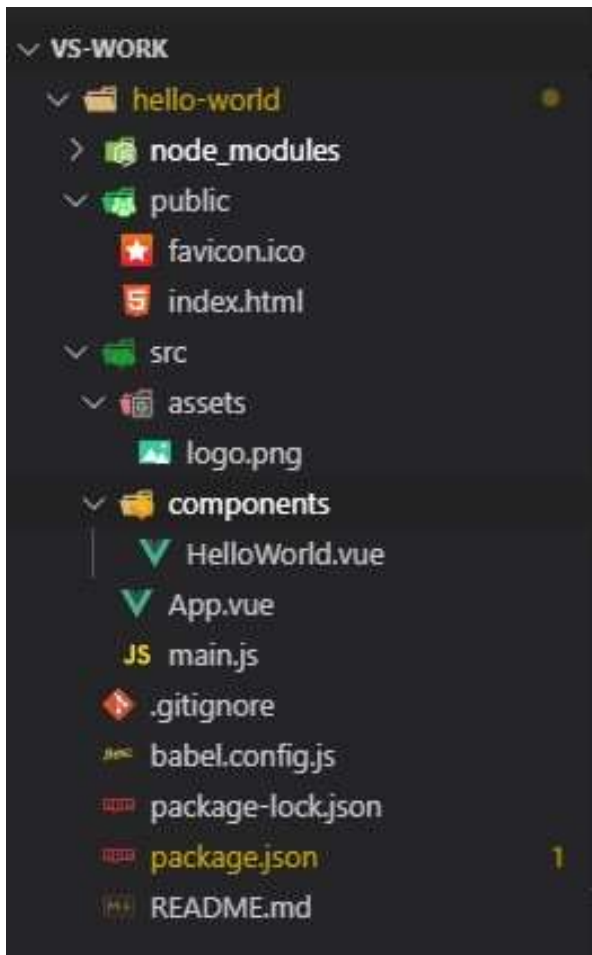
激活 Windows



03 VueCli项目目录解析



• VueCli项目目录解析



- **node_modules**: 项目依赖(内置了webpack), 在当前目录下通过npm安装的非全局依赖库都会放在这个目录下
- **public**: Vue3.x的静态资源目录, 相当于vue-cli2.x中的static, public通常放不会变动的文件, public/ 目录下的文件并不会被Webpack处理: 它们会直接被复制到最终的打包目录(默认是dist/)下。必须使用绝对路径引用这些文件
- **src**:源码目录
 - a) **assets**: Vue3.x的静态资源目录, assets目录中的文件会被webpack处理解析为模块依赖, 只支持相对路径形式,简单来说就是就是public放别人家的资源文件(也就是不会变动, 绝对路径引用), assets放自己的资源文件(需要改动的文件, 相对路径引入)
 - b) **components**: vue组件, 一个Vue组件都对应一个vue实例
 - c) **App.vue**: 默认入口组件, 它的vm实例控制的就是index.html中的div区域
 - d) **main.js**:项目入口文件, webpack编译打包就是从此文件开始, 换言之, 所有你用到的资源, 都应该从这里import引入, 才能被打包进项目。最终打包的js文件会自动插入到index.html页面中
 - e) **babel.config.js**: Babel配置文件, Babel是处理es6高级语法转换库, 它能将es6的很多高级语法转换成低级浏览器可以识别的js代码, 解决兼容性问题, 是项目必备依赖
 - f) **package-lock.json**:记录当前项目依赖的所有库的具体来源和版本号, 锁定版本。
 - g) **package.json**:项目信息说明文件以及依赖包列表, 当拷贝项目源码时, 是必须要忽略node-modules目录的, 其他人要重新安装项目的依赖, 执行npm i, 会根据package.json中指定的依赖列表自动安装到当前项目目录的node-modules中

• VueCli项目目录解析

App.vue组件:

一个vue文件就是一个Vue组件,而这也是开发单页面应用的最核心的技术,每一个vue组件,在vscode安装了相应的辅助插件后,每次创建一个新的vue文件,输入vue,并按一下tab键,就会自动帮我们生产一个vue组件.

```
<template>
  <div id="#app">{{msg}}</div>html
</template>

<script>
export default {
  data () {
    return {
      msg: "这是一个测试vue"
    }
  }
}
</script>
<style scoped>
  div{
    background-color: orange;
  }
</style>
```

- ◆ template标签: 组件模板, 也就是vue组件对应要展示的界面
- ◆ script标签: 定义组件对应的vm实例
- ◆ style: 组件内元素的css样式定义

组件模板

• VueCli项目目录解析

Package.json:

- **package-lock.json**:当执行npm install的时候, node会先从package.json文件中读取所有dependencies信息, 然后根据dependencies中的信息与node_modules中的模块进行对比, 没有的直接下载, node是从package.json文件读取模块名称, 从package-lock.json文件中获取版本号, 然后进行下载或者更新, 当package.json与package-lock.json都不存在, 执行“npm install”时, node会重新生成package-lock.json文件, 然后把node_modules中的模块信息全部记入package-lock.json文件, 但不会生成package.json文件, 这个文件自动维护, 开发者不动
- **package.json**:项目清单文件, 常用选项说明:
 - a) **scripts**:项目运行脚本
 - b) **dependencies与devDependencies**:项目依赖列表, dependencies是开发和正式生产环境都需要的依赖, 而devDependencies表示开发依赖, 区别就在于, 当项目被webpack打包编译后, devDependencies包含的依赖库不会被打包进去
 - c) **eslintConfig**:eslint是一个插件化的javascript代码检测工具, eslintConfig是默认vue-cli3.x对于vue项目启动的eslint配置



• VueCli项目目录解析

Package.json的常用选项示例说明:scripts

```
"scripts": {  
  "serve": "vue-cli-service serve --open",  
  "build": "vue-cli-service build",  
  "lint": "vue-cli-service lint"  
}
```

- **serve**: 对应终端下 输入npm run serve命令, 实际执行vue-cli-service serve - open命令, 表示运行当前项目, --open选项表示自动打开浏览器, 具体命令配置, 参考:
<https://cli.vuejs.org/zh/guide/cli-service.html#使用命令>。
- **build**: 对应终端下输入npm run build, 实际执行vue-cli-service build命令, 表示执行正式环境的编译打包, 会生成dist目录, 因为此命令是启用内置webpack进行资源编译打包输出。
- **lint**: 对应npm run lint, 启用eslint插件进行语法检查, 不会启动项目, 也不会编译项目, 如果js语法不符合规范, 控制台会抛出异常



• VueCli项目目录解析

Package. json的常用选项示例说明: **eslintConfig**

```
"eslintConfig": {
```

“**root**”: **true**, //ESLint 会在所有父级目录里寻找配置文件，一直到根目录。如果发现配置文件中 “**root**”: **true**，它就会停止在父级目录中寻找,减少检查时间,提高效率

```
  "env": {
```

“**node**”: **true** //启用eslint的环境, 可选配置有 **es6**: **true** (支持es6高级语法), **node**: **true**(支持node语法), **browser**: **true**,
jquery: **true**

```
  },
```

```
  "extends": [
```

```
    "plugin:vue/essential",
```

```
    "eslint:recommended" //启动推荐规则
```

```
  ],
```

“**rules**”: {}, //具体检测规则，比如: 'indent': [2, 4], // 强制使用一致的缩进, 'semi': [2, 'never'], // 要求或禁止使用分号代替ASI, 数字中的**2**表示开启规则，**error**级别，一旦出现问题，程序会退出，**1**: 启用规则，**warn**级别，一旦出现问题，程序不会退出，**0**: 关闭规则

```
  "parserOptions": {
```

```
    "parser": "babel-eslint" //指定解析器，eslint默认用Espre解析器，这里指定了babel-eslint，兼容espress.
```

```
  }
```

```
}
```

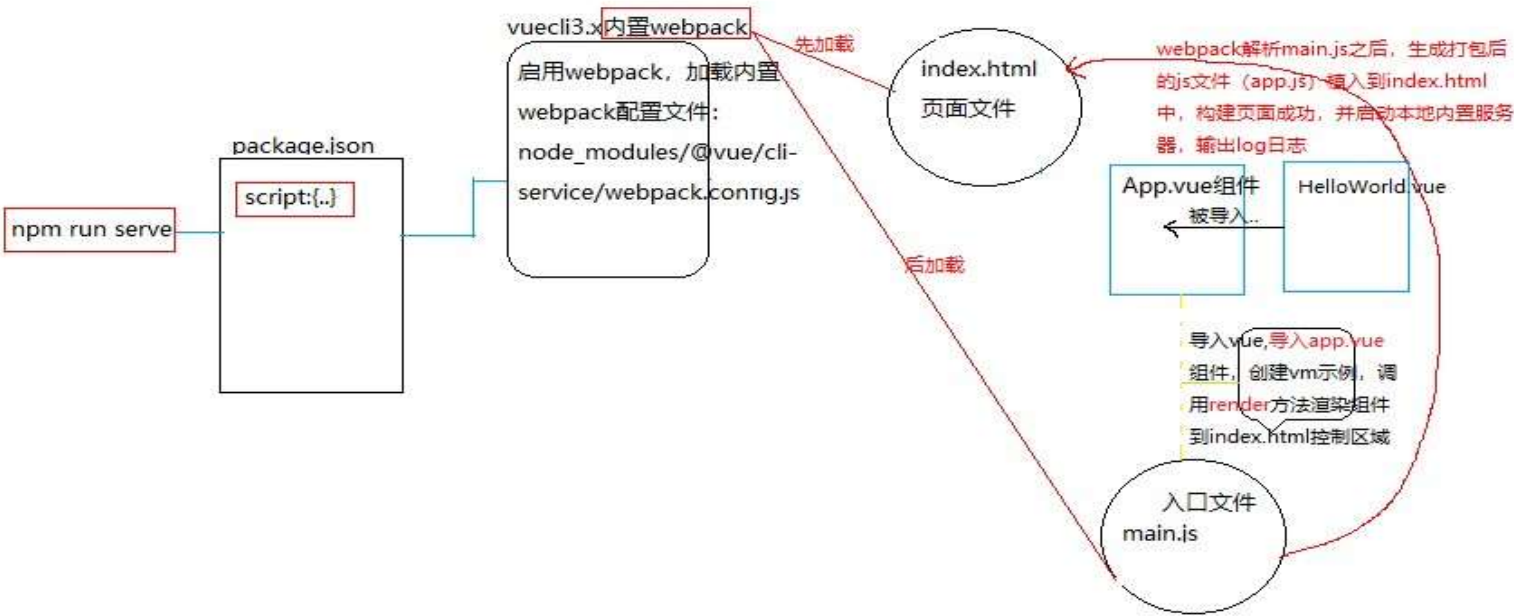


04 VueCli项目启动过程



• VueCli项目目录解析

VueCli3.x默认创建的Vue项目启动过程，自绘了一张简要的图，进行说明：



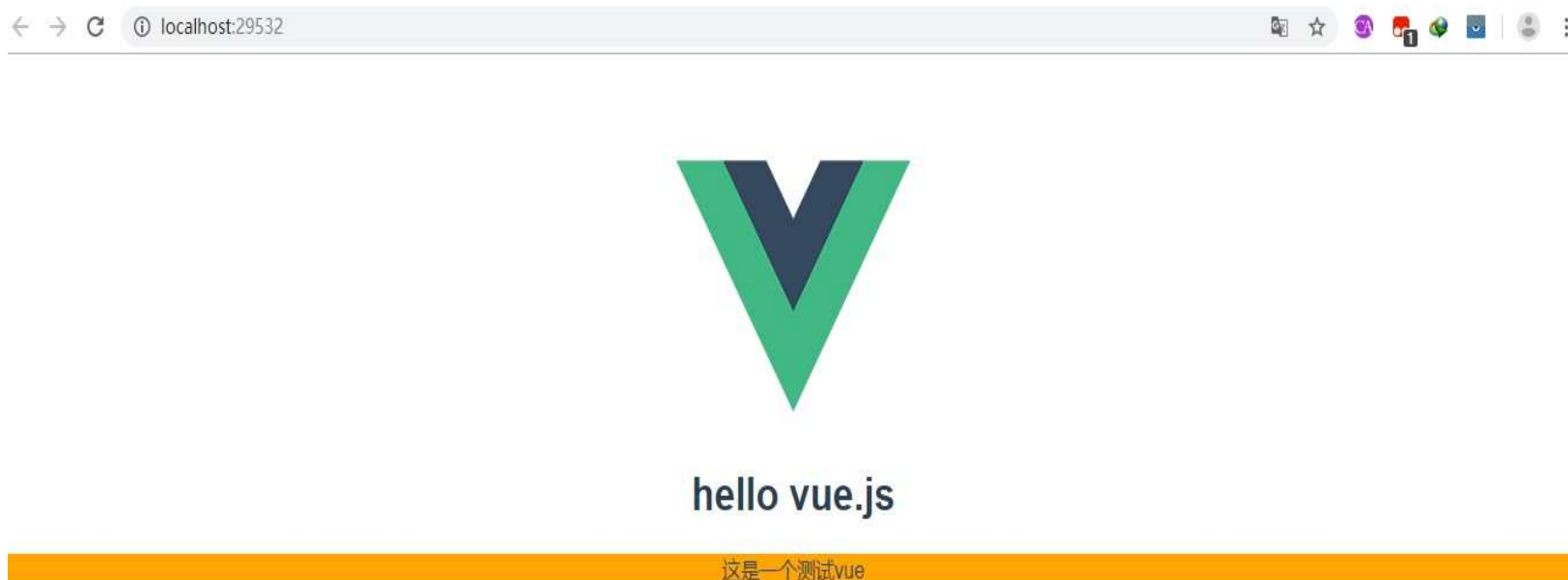
• 本节总结

- ◆ SPA认识
- ◆ Vue的数据驱动和组件化理解：MVVM架构
- ◆ VueCli3.x项目目录认识
- ◆ VueCli3.x项目启动过程



• 本节练习

◆ 作业1：根据自己的理解，更改默认创建的VueCli3.x项目代码，将项目启动后的默认界面展示如下信息



THANK YOU



软通大学
ISOFTSTONE UNIVERSITY