

## 1:先搭建一个项目目录

dist

css

image

src

| index.html

| main.js

## 2:初始化这个项目

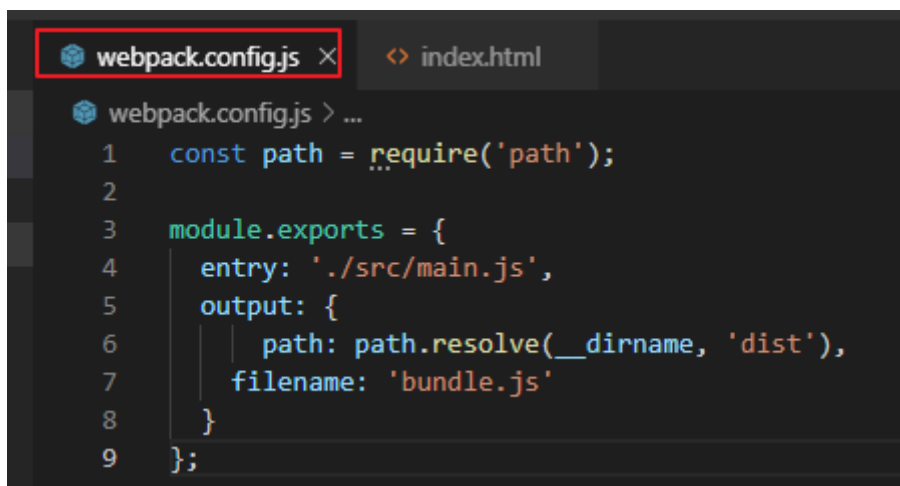
npm init -y 即把这个项目交给webpack托管

## 3:安装本项目的webpack包

cnpm install --save-dev webpack

cnpm i -D webpack-cli

配置

A screenshot of a code editor with two tabs: 'webpack.config.js' (active) and 'index.html'. The 'webpack.config.js' file contains the following code:

```
1  const path = require('path');
2
3  module.exports = {
4    entry: './src/main.js',
5    output: {
6      path: path.resolve(__dirname, 'dist'),
7      filename: 'bundle.js'
8    }
9  };
```

## 4: 安装自动化打包工具

cnpm i webpack-dev-server -D

A screenshot of a code editor with three tabs: 'webpack.config.js', 'index.html', and 'package.json' (active). The 'package.json' file contains the following code:

```
4  "description": "",
5  "main": "webpack.config.js",
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1",
8    "dev": "webpack-dev-server --open --port 3000 --contentBase src --hot"
9  },
```

## 5: 在普通页面中使用vue是先用script src引包

然后 `new Vue({});` 创建vue实例

在页面中创建vue 控制的id为app的容器

所以在webpack中，使用vue，需要先使用命令来导入包

```
npm i vue -S
```

```
42   render: c => c(login)
43 })
44
45
46 // 总结梳理：webpack 中如何使用 vue：
47 // 1. 安装vue的包： cnpm i vue -S
48 // 2. 由于 在 webpack 中，推荐使用 .vue 这个组件模板文件定义组件，所以，需要安装 能解析这种文
    件的 loader    cnpm i vue-loader vue-template-compiler -D
49 // 3. 在 main.js 中，导入 vue 模块  import Vue from 'vue'
50 // 4. 定义一个 .vue 结尾的组件，其中，组件有三部分组成： template script style
51 // 5. 使用 import login from './login.vue' 导入这个组件
52 // 6. 创建 vm 的实例 var vm = new Vue({ el: '#app', render: c => c(login) })
53 // 7. 在页面中创建一个 id 为 app 的 div 元素，作为我们 vm 实例要控制的区域；
```

但是 使用 `import Vue from 'vue'` 这种写法的话，

```
webpack.config.js  JS main.js  ×  <> index.html
src > JS main.js > [e]vm
1  | import Vue from 'vue'
2  | // import './css/index.css'
3
4  | console.log('hello-webpack-vue');
5  | var vm = new Vue({
6  |   el: "#app",
7  |   data: {
8  |     msg: "123"
9  |   },
10 |   methods: {}
11 | };
12
```

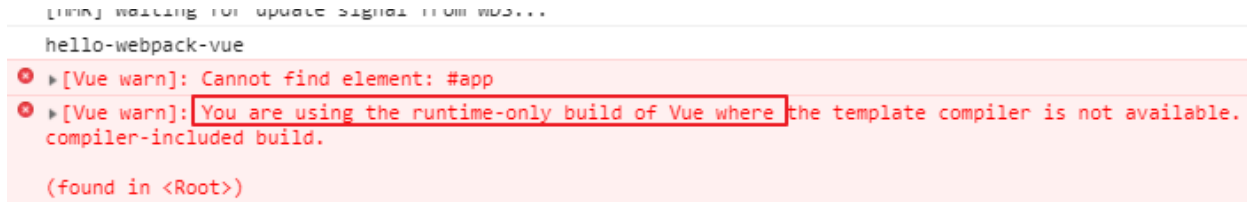
会报错，因为这个包不全，有残缺，只导入了 `runtime-only`的方式，并没有提供像网页中那样的使用方式

## 5.1 包的查找规则，

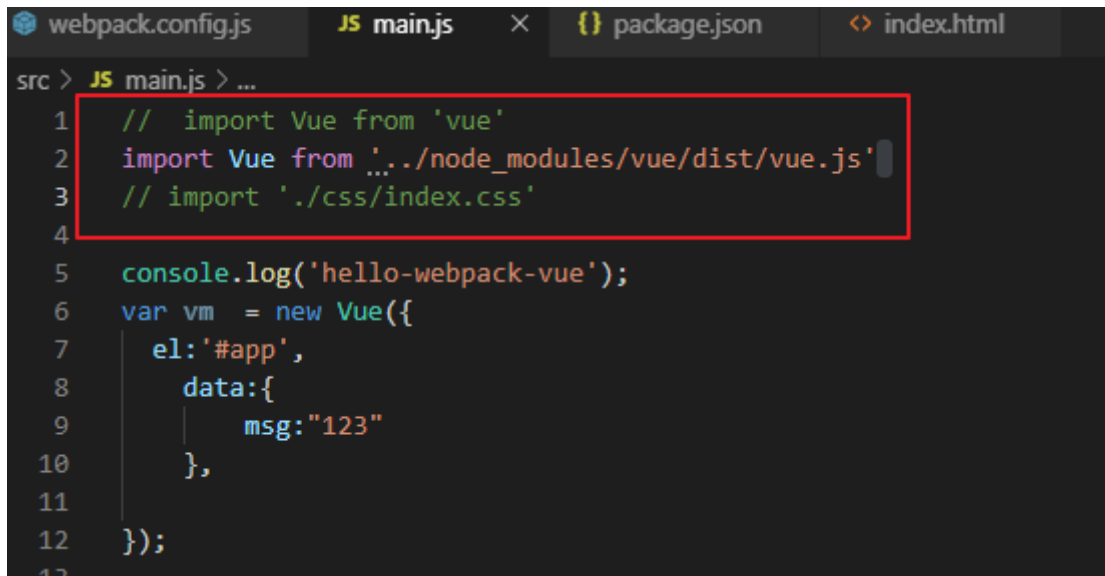
- 1: 找项目根目录的 `node_module`的文件夹
- 2: 在`node_module`中，根据报名，找对应的vue文件夹
- 3: 在vue文件夹中，找一个叫做`package. json`的包配置文件

4: 在package.json文件夹中，查找一个main属性【main属性指定了这个】包被加载的时候的入口文件

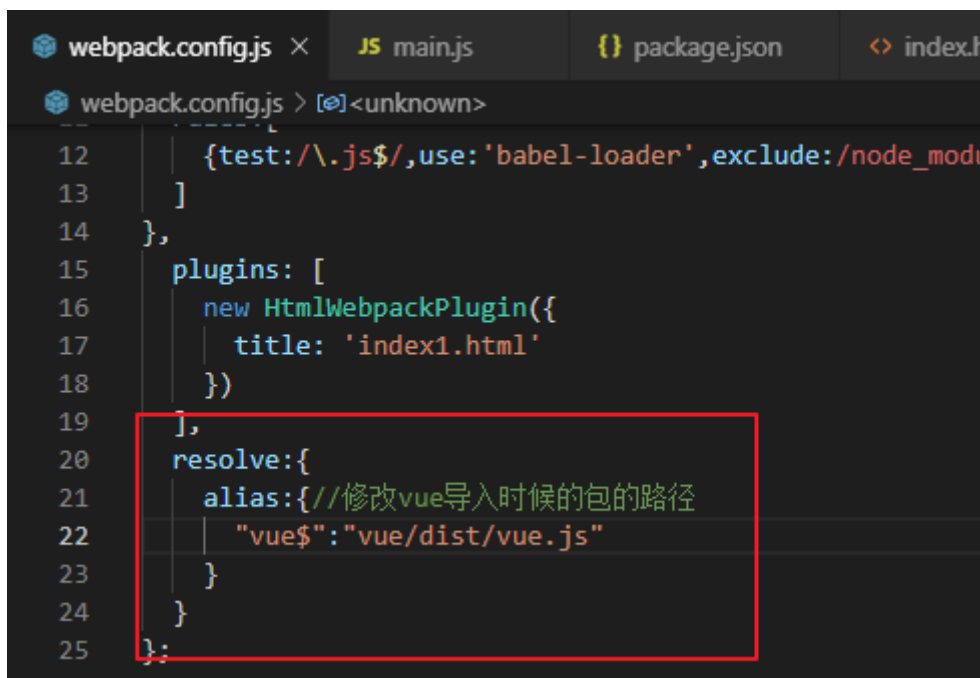
要解决这个问题:



方法一： 所以导入包的时候，导入如下的包可解决这个问题



方法二： 在webpack.config.js中，与module同级，



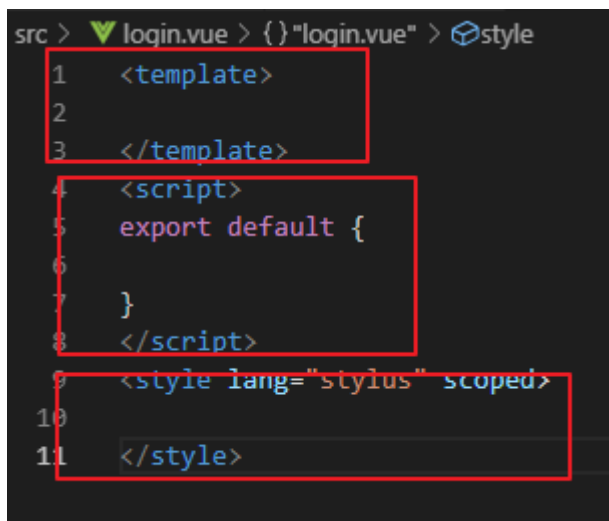
同时，main.js文件中还是

```
import Vue from 'vue'
```

修改配置文件需要重新启动项目

## 6: .vue文件（专门的组件文件）

6.1在src文件下，新建 login.vue文件



```
src > login.vue > {} "login.vue" > style
1  <template>
2
3  </template>
4  <script>
5    export default {
6
7    }
8  </script>
9  <style lang="stylus" scoped>
10
11 </style>
```

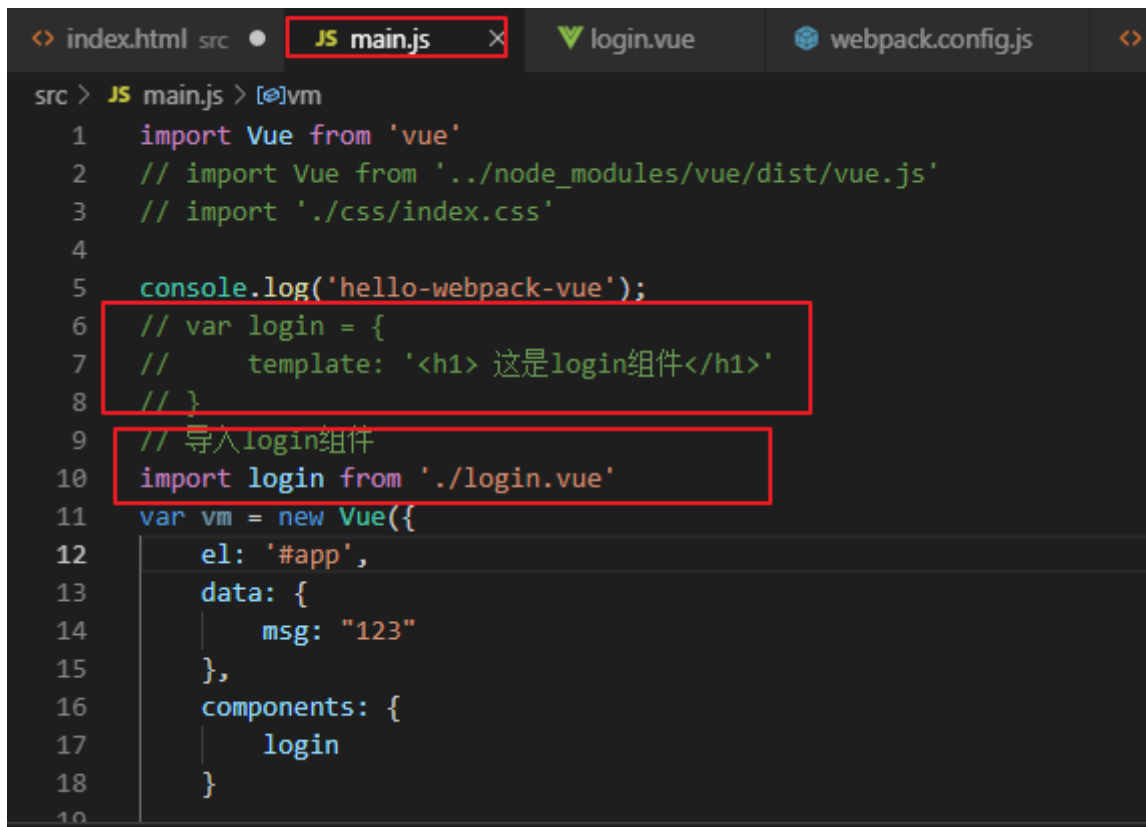
<template>中写html代

<script>中写业务逻辑代码

<style>中写样式代码

6.2导入.vue的组件

可以替代写在js文件中的组件模板对象

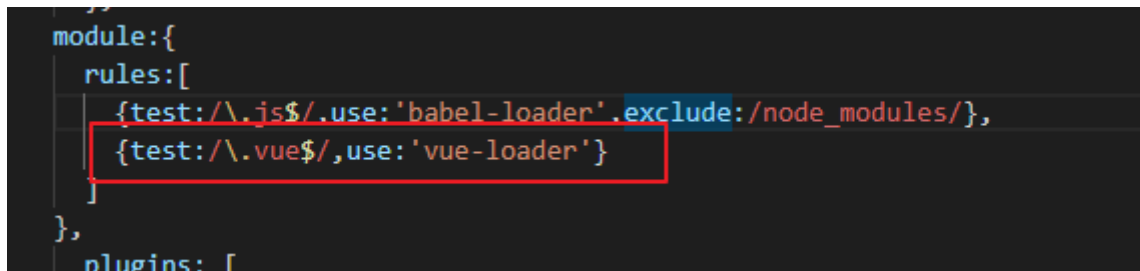


```
src > JS main.js > [vm]
1  import Vue from 'vue'
2  // import Vue from '../node_modules/vue/dist/vue.js'
3  // import './css/index.css'
4
5  console.log('hello-webpack-vue');
6  // var login = {
7  //   template: '<h1> 这是login组件</h1>'
8  // }
9  // 导入login组件
10 import login from './login.vue'
11 var vm = new Vue({
12   el: '#app',
13   data: {
14     msg: "123"
15   },
16   components: {
17     login
18   }
19 })
```

默认，webpack 无法打包 .vue 文件，需要安装 相关的loader：

cnpm i vue-loader vue-template-compiler -D

在配置文件中，新增loader配置项 { test: /\.vue\$/, use: 'vue-loader' }



```
module:{
  rules:[
    {test:/\.js$/,use:'babel-loader'.exclude:/node_modules/},
    {test:/\.vue$/,use:'vue-loader'}
  ]
},
plugins: [
```

### 6.3: 渲染组件

在 webpack 中，如果想要通过 vue， 把一个组件放到页面中去展示，vm 实例中的 render 函数可以实现

```

9 // 导入login组件
10 import login from './login.vue'
11 var vm = new Vue({
12   el: '#app',
13   data: {
14     msg: "123"
15   },
16   // components: {
17   //   login
18   // }
19   render:function(c){
20     return c(login);
21   }
22 });

```

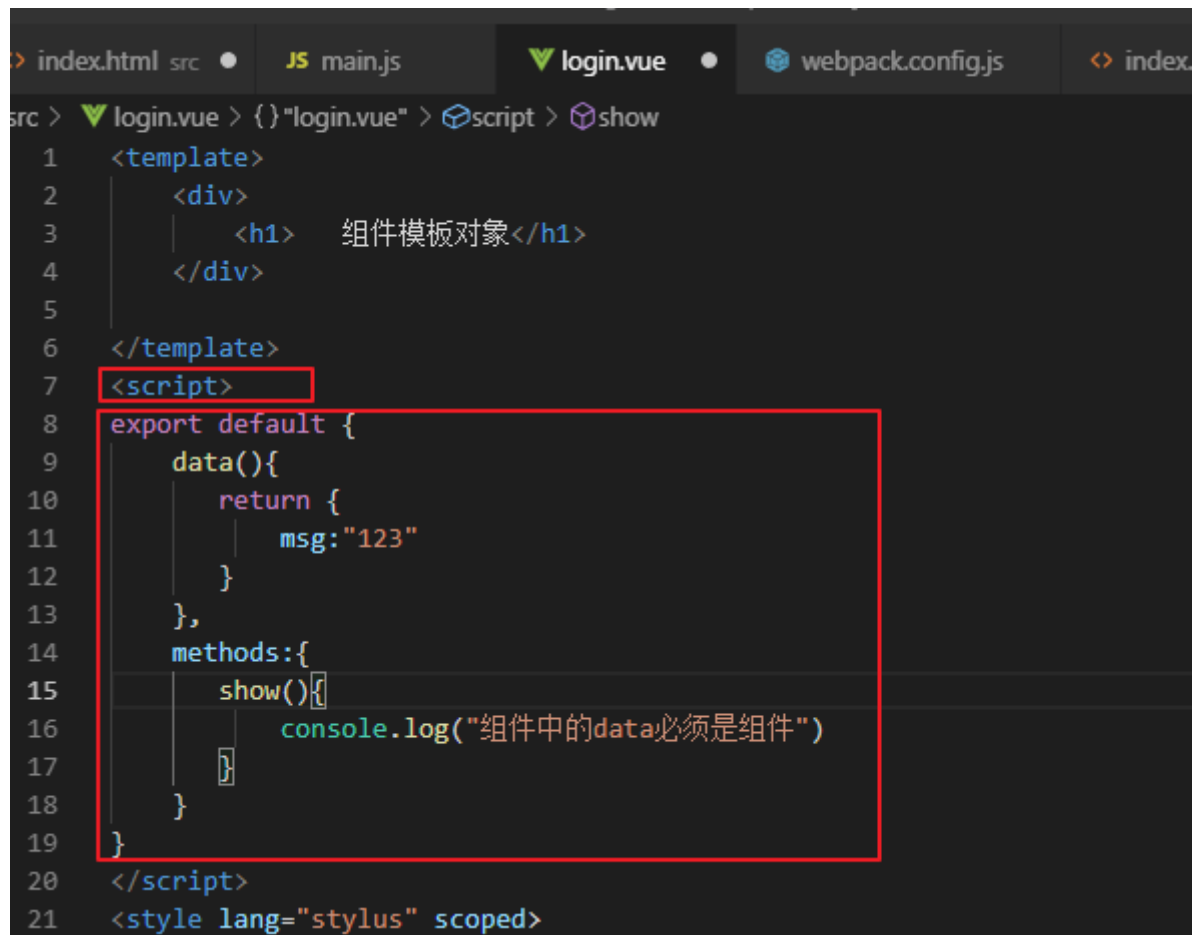
## 7: 总结:

```

15
16 // 总结梳理：webpack 中如何使用 vue：
17 // 1. 安装vue的包： cnpm i vue -S
18 // 2. 由于 在 webpack 中，推荐使用 .vue 这个组件模板文件定义组件，所以，需要安装 能解析这种文
   件的 loader    cnpm i vue-loader vue-template-compiler -D
19 // 3. 在 main.js 中，导入 vue 模块 import Vue from 'vue'
20 // 4. 定义一个 .vue 结尾的组件，其中，组件有三部分组成： template script style
21 // 5. 使用 import login from './login.vue' 导入这个组件
22 // 6. 创建 vm 的实例 var vm = new Vue({ el: '#app', render: c => c(login) })
23 // 7. 在页面中创建一个 id 为 app 的 div 元素，作为我们 vm 实例要控制的区域；

```

## 8: 在组件中给组件定义方法



```
src > login.vue > {} "login.vue" > script > show
1  <template>
2    <div>
3      <h1> 组件模板对象</h1>
4    </div>
5
6  </template>
7  <script>
8    export default {
9      data(){
10        return {
11          msg:"123"
12        }
13      },
14      methods:{
15        show(){
16          console.log("组件中的data必须是组件")
17        }
18      }
19    }
20  </script>
21  <style lang="stylus" scoped>
```

注意: 组件中的data必须是方法

## 8.1 两种暴露和导入模块的方法，不可混用

node中向外暴露成员的方法

```
module.exports = {}
```

用 `var 名称 = require("模块标识符");` 导入

在es6的新语法中，使用 `export default {}` 和 `export` 向外暴露成员

使用 `import 模块名称 from "模块标识符"` 来导入模块

例如:

暴露时:

```
export default {
  name:"lisi",
  age:20
}
```

接受时:

```
import ml from "哪个暴露对象的文件地址"
```

m1: 用于接收的变量名，可用任意名字来接受

在一个模块中，export default只允许向外暴露一次

```
// 在 ES6 中，使用 export default 和 export 向外暴露成员：
var info = {
  name: 'zs',
  age: 20
}

export default info

// 注意： export default 向外暴露的成员，可以使用任意的变量来接收
```

在一个模块中，可以同时使用export 和export default 暴露成员，这样就可以暴露多个成员

export 可以暴露多个成员

但是 使用 export 暴露的成员，只能使用花括号 导出，学术名叫（按需到处）如下：

```
export var title = '小星星'
export var content = '哈哈'
```

```
import m222, { title, content } from './test.js'
console.log(m222)
console.log(title + ' --- ' + content)
```

也可以导出时其别名

```
import m222, { title as title123, content } from './test.js'
console.log(m222)
console.log(title123 + ' --- ' + content)
```