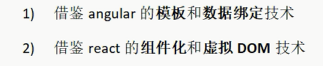
1. 简介

渐进式JavaScript 框架。可以只用核心库，当需要特定的功能时，添加相应组件。REACT也是这样的。





1. 官网

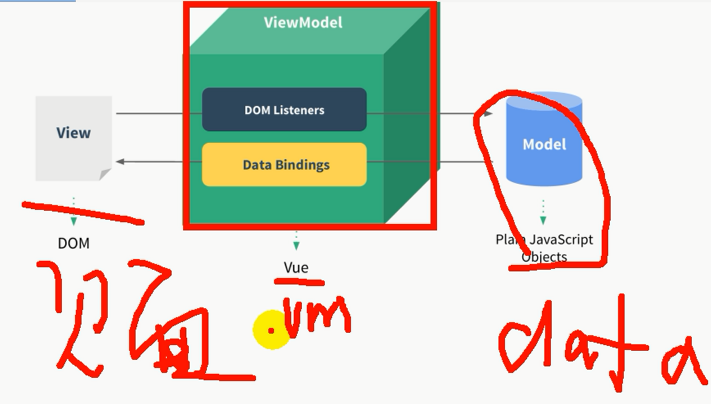
https://cn.vuejs.org/

1. MVVM

model+view+viewmodel(view的实例)

这里的data可以写对象，也可以写函数





* 1. view:里面有指令（自定义标签属性）和大括号表达式（用来显示数据）
  2. model:数据
  3. vm:数据绑定和数据监听

1. Idea提示

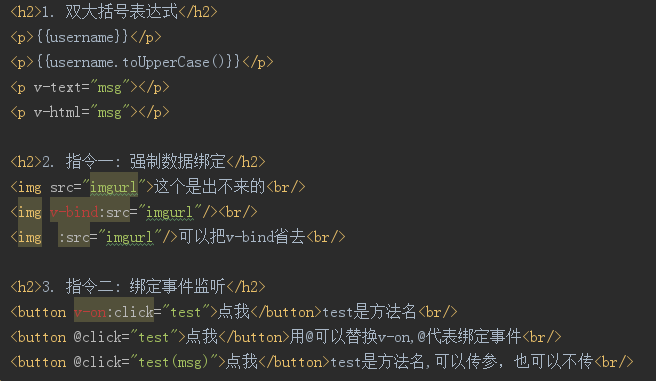
http://www.cnblogs.com/lovling/p/9339871.html

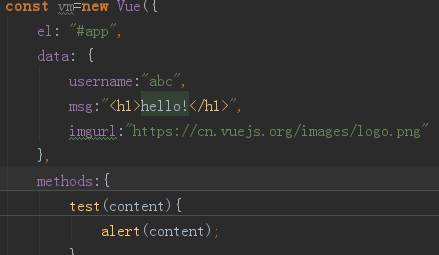
1. chrome插件

vue-devtools-4.1.4\_0.crx

1. 模板语法

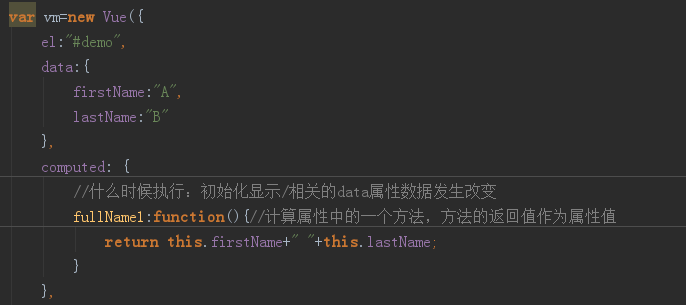
1. 模板的理解:  
 动态的html页面  
 包含了一些JS语法代码  
 大括号表达式  
 指令(以v-开头的自定义标签属性)  
2. 双大括号表达式  
 语法: {{exp}} 或 {{{exp}}}  
 功能: 向页面输出数据  
 可以调用对象的方法  
3. 指令一: 强制数据绑定  
 功能: 指定变化的属性值  
 完整写法:  
 v-bind:xxx='yyy' //yyy会作为表达式解析执行  
 简洁写法:  
 :xxx='yyy'  
4. 指令二: 绑定事件监听  
 功能: 绑定指定事件名的回调函数  
 完整写法:  
 v-on:click='xxx'  
 简洁写法:  
 @click='xxx'





1. 计算属性和监视

1. 计算属性（用的特别多）  
 在computed属性对象中定义计算属性的方法  
 在页面中使用{{方法名}}来显示计算的结果  
2. 监视属性:(用的少一些)  
 通过通过vm对象的$watch()或watch配置来监视指定的属性  
 当属性变化时, 回调函数自动调用, 在函数内部进行计算  
3. 计算属性高级:  
 通过getter/setter实现对属性数据的显示和监视  
 计算属性存在缓存, 多次读取只执行一次getter计算







1. class与style绑定

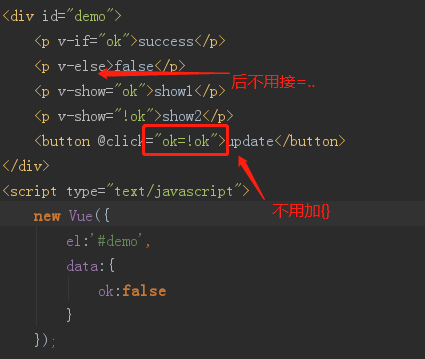
1. 理解  
 在应用界面中, 某个(些)元素的样式是变化的  
 class/style绑定就是专门用来实现动态样式效果的技术  
2. class绑定: :class='xxx'  
 xxx是字符串  
 xxx是对象  
 xxx是数组（不常用）  
3. style绑定（必须用对象来绑定）  
 :style="{ color: activeColor, fontSize: fontSize + 'px' }"  
 其中activeColor/fontSize是data属性



1. 条件渲染

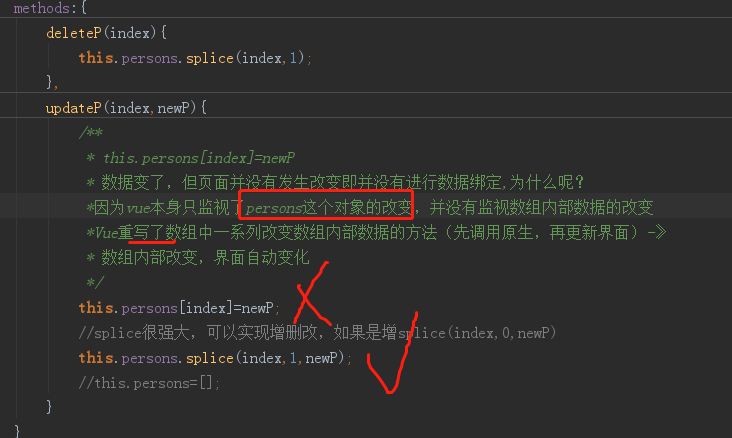
v-if,v-else 条件判断

v-show：显示隐藏

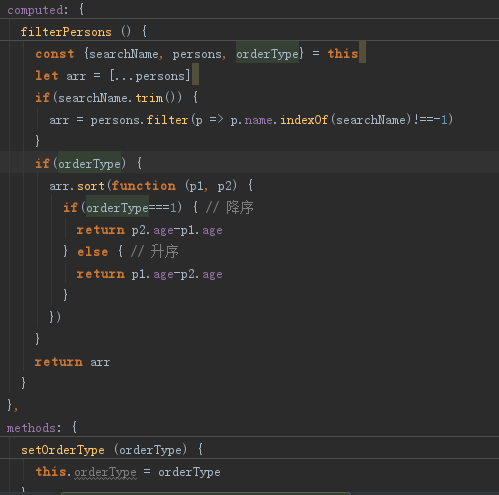


1. 列表渲染
   1. 列表遍历

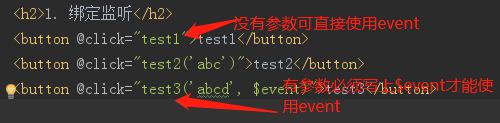


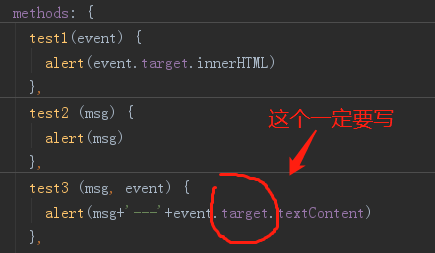


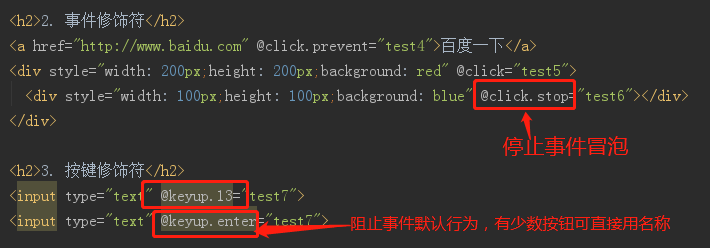
* 1. 列表过滤排序



1. 事件处理

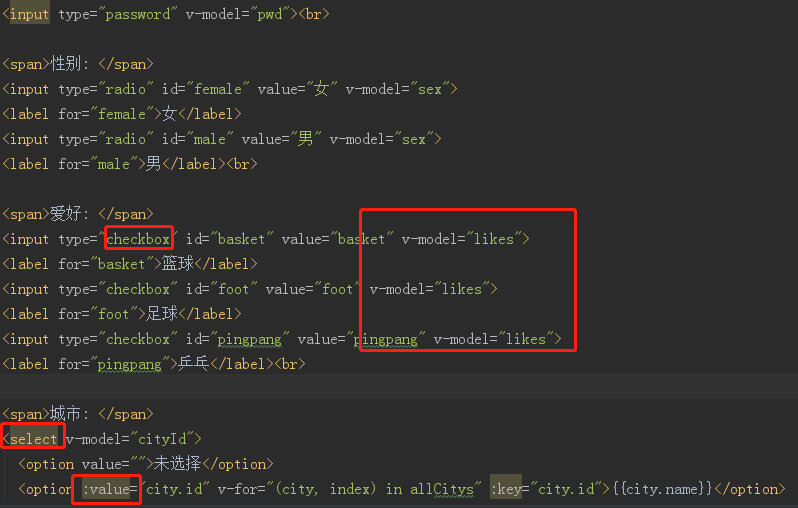






1. 表单输入绑定

用v-model



1548137604(1)

1. Vue实例\_生命周期



1. 过渡&动画

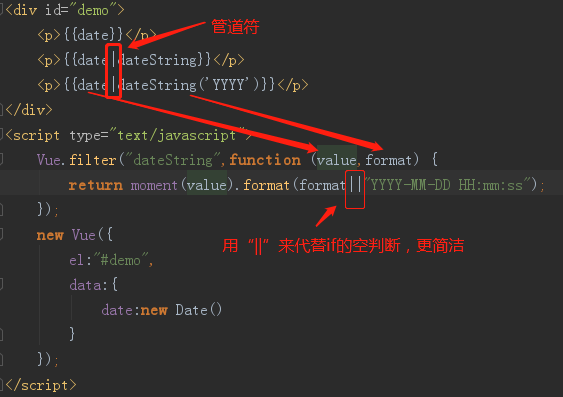
见官网

1. 过滤器

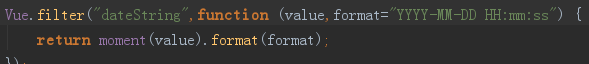




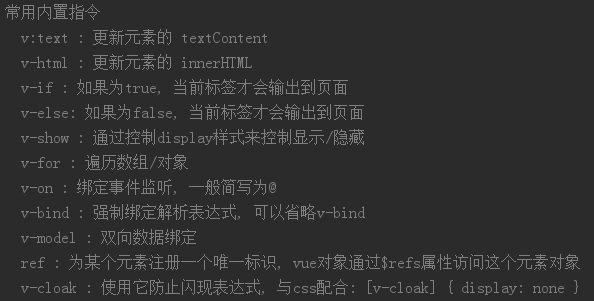
以date格式化来举例。可以去bootcdn里去找到moment.js



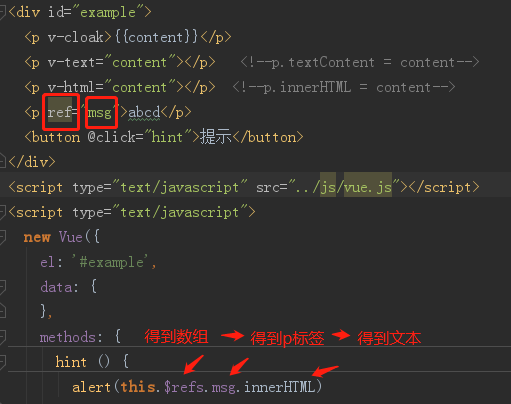
下面这样也可以



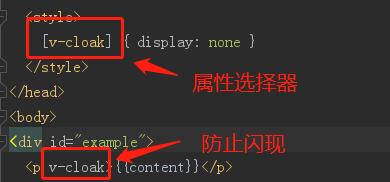
1. 指令
   1. 常用内置指令



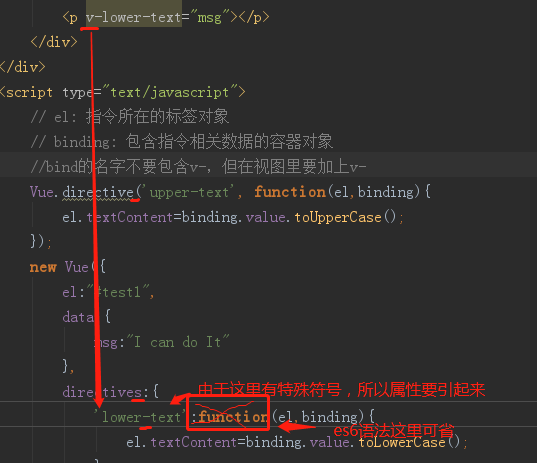
ref



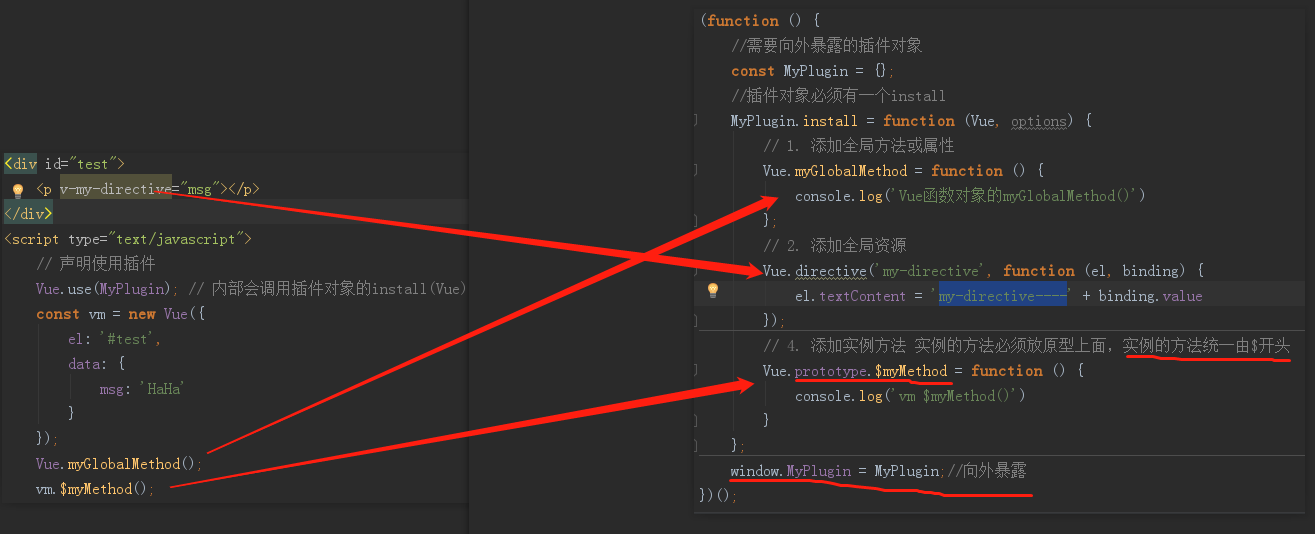
v-clock



* 1. 自定义指令



1. 插件

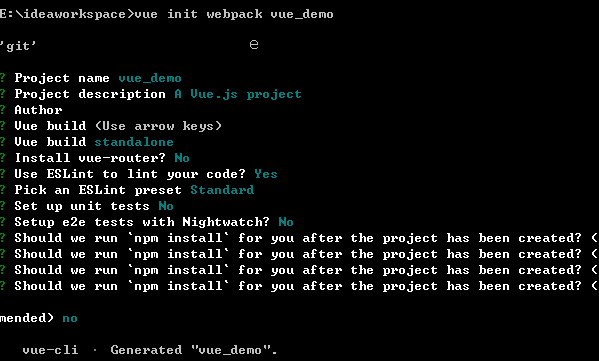


1. vue-cli

是vue官方提供的脚手架

npm install -g vue -cli

vue init webpack vue\_demo

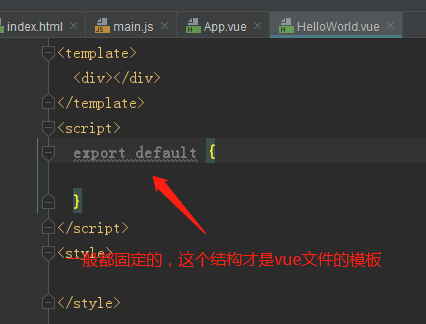
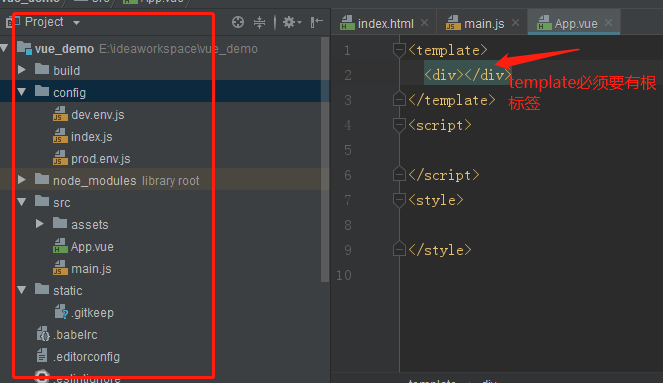


cd vue\_demo

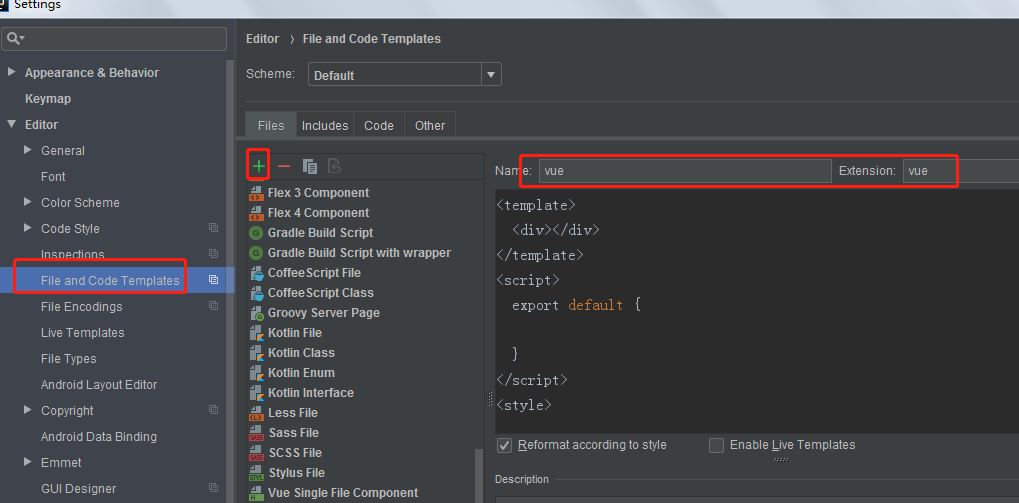
npm install npm@latest -g

npm run dev或者npm start

自动创建的目录结构



所以可以新建模板



这样就可以直接右键new-vue了。

开始搭建脚手架

1.编写组件



1. 编写主组件



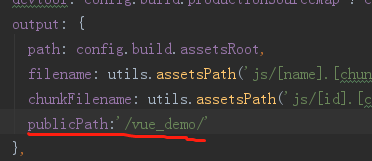
1. 写JS（下面是基本固定的写法）



1. 打包发布项目

执行npm run build，就会生成一个dist文件夹，里面的内容就是打包后的文件

* 1. 使用静态服务器工具包
     1. npm install -g serve
     2. serve dist
     3. 访问http:localhost:5000
  2. 使用动态web服务器(tomcat)
     1. 修改配置webpack.prod.conf.js，加上项目名



* + 1. 重新打包npm run build
    2. 把dist文件COPY出来，改名为项目名，放到TOMCAT下，启动TOMCAT访问

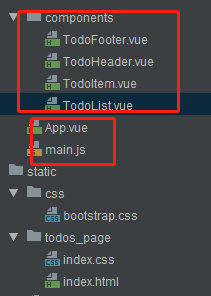
1. eslint代码检查工具

.eslintrc.js可以配置规则

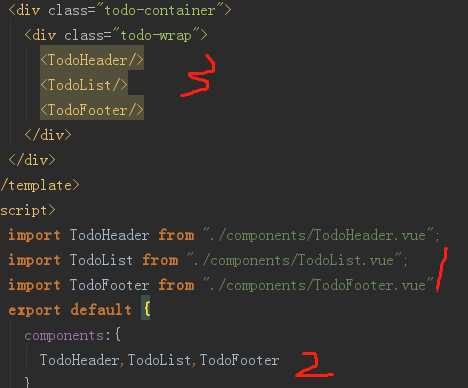
1. 项目实例



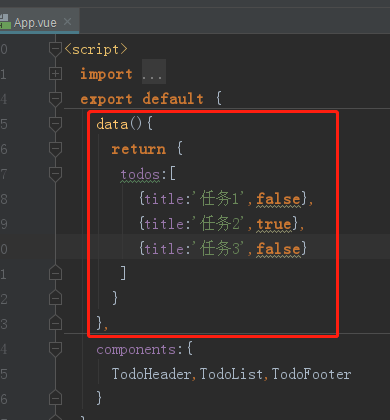
* 1. 搭出框架
  2. 拆分组件
     1. 把不同的HTML都放到对应的VUE中



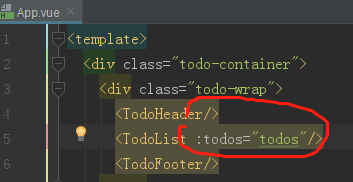
* + 1. 在App.vue中引入需要用到的vue,定义组件，并在模板中写入标签



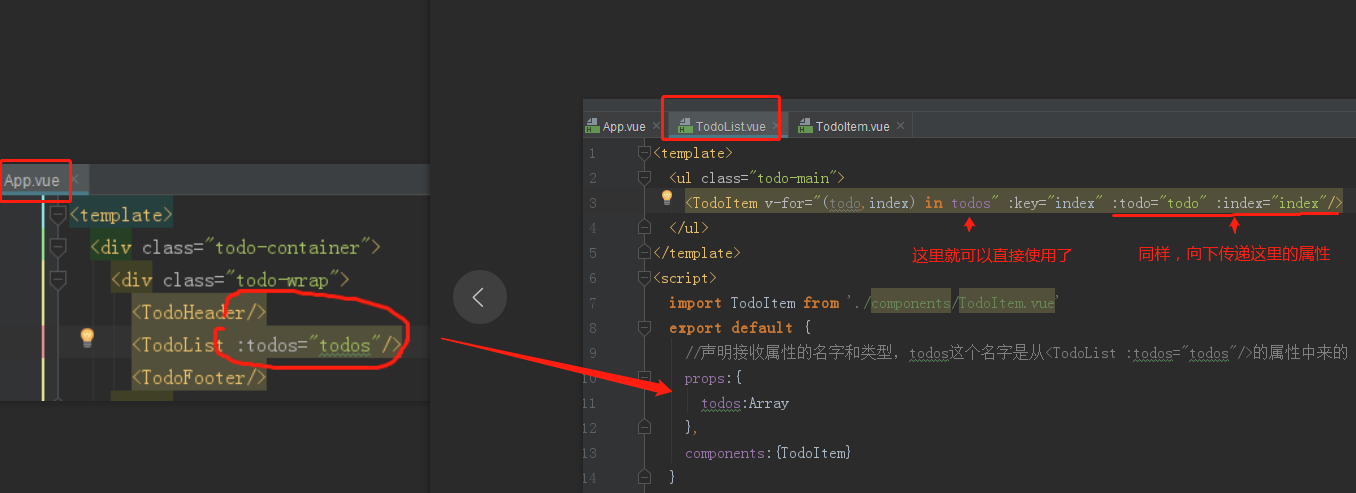
* + 1. 拆分css,如果有公用的可以抽取出来放到base.css,main.js来引用import './base.css',静态页面到此完成
    2. 分析组件内需要动态变化的是哪些数据，然后把这些数据分析成对象，并确定对象的属性和类型。再分析这些数据分别放在哪个组件里去保存。



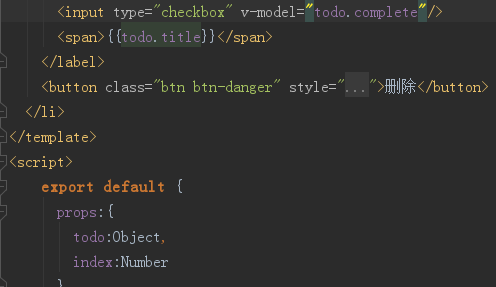
* + 1. 初始化数据谁去显示?list组件去显示，然后去list组件中开发



这里:todos=”todos”的意思是：向TodoList组件传递了在App.vue这里定义的todo属性

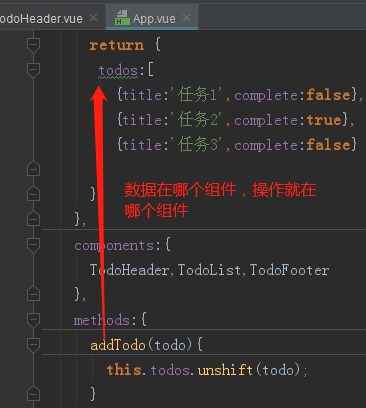
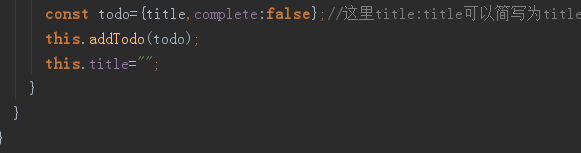


写item

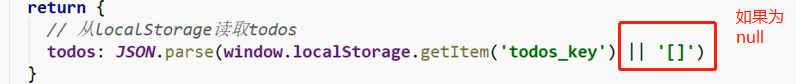


* + 1. 交互需要先绑定监听（HTML中写事件），然后写vue的data和methods.（绑定v-model不要忘记）并且，**数据在哪个组件，更新这个数据的行为就应该在哪个组件**





* 1. 存储数据
     1. 对数据读取



* + 1. 对数据监视

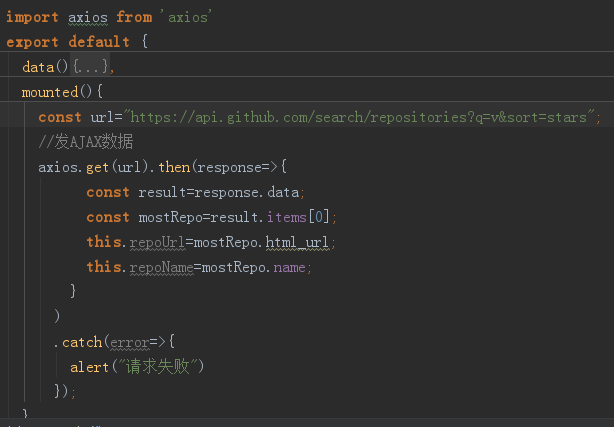
需要深度监视。一般监视只能监视最外层的



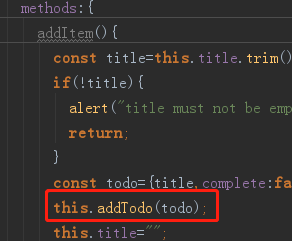
1. Ajax

axios：通用的ajax请求库，官方推荐

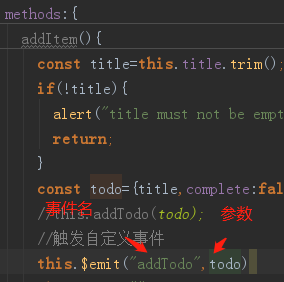
npm install axios --save



1. 组件间通信
   1. prop（见前面）

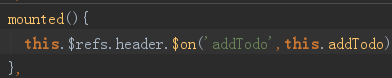
1548294604(1)1548294626(1)

* 1. 自定义事件

1548294917(1)（专门用于父子关系传递）

如果不在图1上写@addTodo=”addTodo”的话，可以在App的mounted里做操作

this.$on(“addTodo”,this.addTodo)。但是这里的this是APP，所以需要这样做

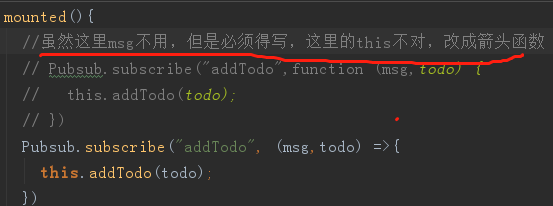
1548295349(1)

* 1. 消息订阅，对于传递的两个组件没有任何位置上的要求

首先需要安装 npm install --save pubsub-js

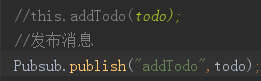
订阅消息：

1548295747(1)



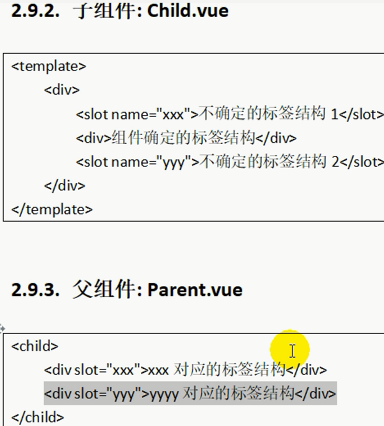
发布消息：

1548295747(1)

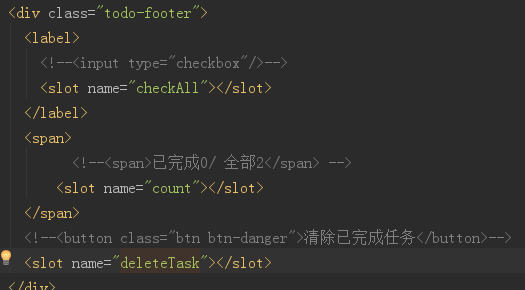


* 1. slot 组件如果需要反复使用的话会使用这个

slot通信的是标签，其他（props,自定义事件，消息订阅）通信的是数据



子：



父：



1. 组件库

Mint UI和Element

1. 路由连接

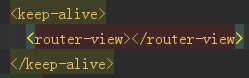
不向后台发请求,路由：键（path）值(组件或处理请求的回调函数)映射关系

vue-router

文档：<https://router.vuejs.org/zh-cn>

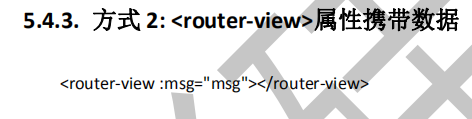
下载：npm install vue-router --save

* 1. 相关说明
     1. VueRouter()用于创建路由器的构建函数
     2. routes:[{path:...,components...}]路由配置
     3. 嵌套路由的话routes:[{path:...,components...,children:[{path:..,components}]}]
     4. 注册路由器 new Vue({router})
     5. 使用路由组件标签<router-link >用来生成路由连接 <router-view>用来显示当前路由组件界面
     6. 缓存路由组件 切换标签时另一个TAB的数据会消失，如果需要保持可以缓存组件对象



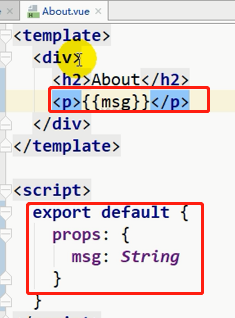
* + 1. 路由路径携带参数





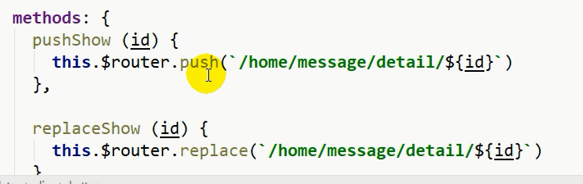
区别只是上面msg是变量，下面abc是字符串

1548399726(1)



* + 1. 编程式路由导航

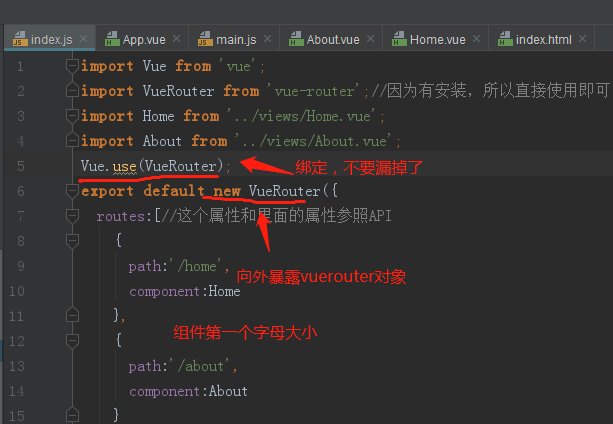
$route代表当前组件，$router代表路由器



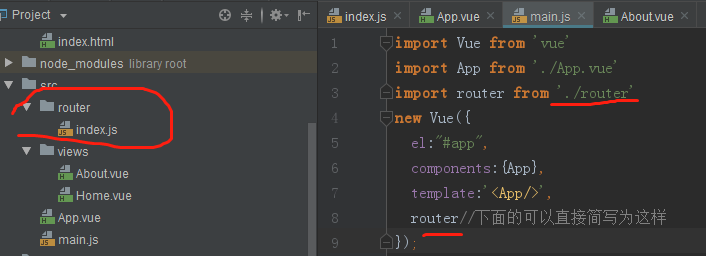


* 1. 基本路由实例

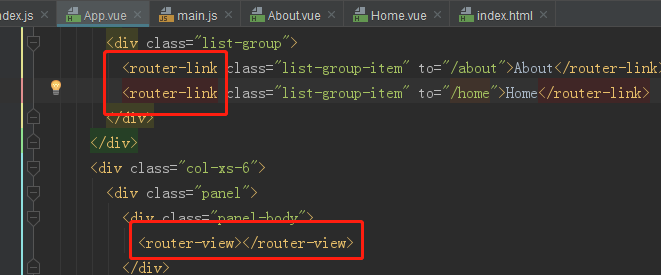
路由：



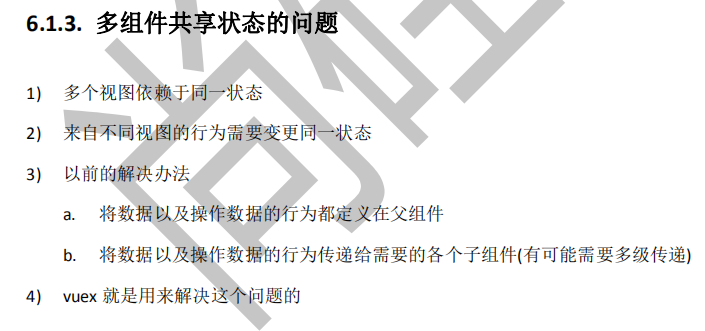
路由器：



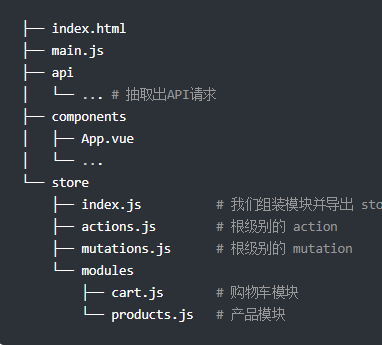
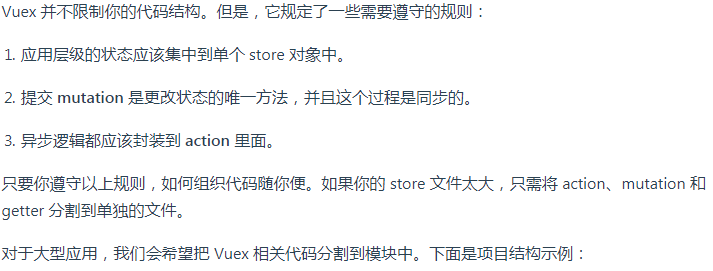
界面：



1. VUEX
   1. 说明 vuex是vue的一个插件 对vue应用中多个组件的共享状态进行集中式的管理（读/写）
   2. 文档：<https://vuex.vuejs.org/zh/guide/>
   3. 产生背景



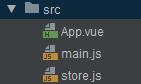
* 1. 安装 npm install --save vuex
  2. 项目结构

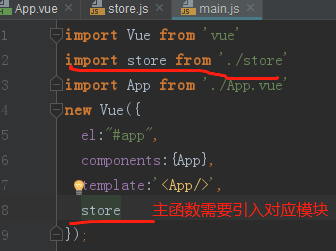


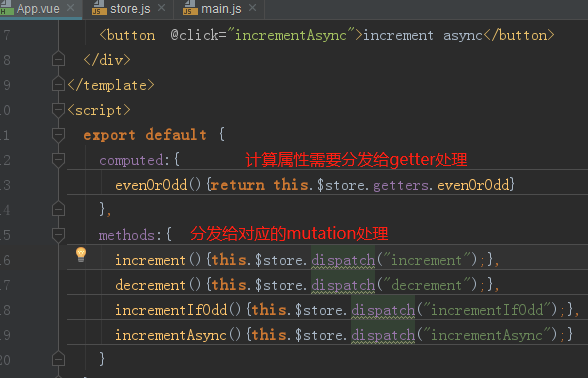
所有用vuex管理的组件中都多了一个属性$store,它就是一个store对象

* 1. 核心概念
     1. store:vuex管理的状态对象，他应该是**唯一**的
     2. mutations
        1. 更改状态的唯一方法，只能包含同步代码，不能异步。
        2. 谁来触发：action中的commit(‘mutation名称’)
     3. actions
        1. 异步逻辑
        2. 谁来触发：组件中$store.dispatch(‘action名称’,data1)
     4. getters
        1. 包含多个计算属性(get)的对象
        2. 谁来读取 组件中$store.getters.xxx
     5. modules 包含多个module,一个module是一个store的配置对象，并且与一个组件对应
  2. 代码实现

这里没有多个module,就直接用store名字来取代

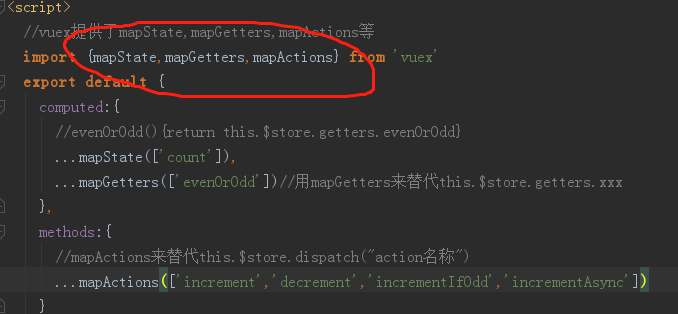








优化App.vue组件



1. VUEX综合案例

actions:包含多个由组件（dispatch写在组件里）触发mutation调用间接更新状态的方法（都是方法）的对象

mutations:包含多个由action（commit写在组件里）触发直接更新状态的方法（都是方法）的对象

getters:包含所有基于state的getter的计算属性的对象

state:把所有公用属性抽取出来

