**Vue进阶**

课程介绍

## Vue基础回顾

## Vue-cli（掌握）

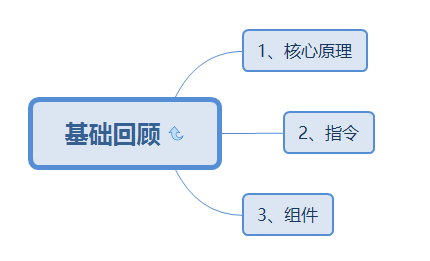
## 生命周期（掌握）

## 计算属性computed

## 组件通信（掌握）

## Vue-router ( 掌握 )

# Vue回顾



## Vue环境搭建

搭建vue项目3大步：

1. 创建视图容器

2. 引入核心库

3. 创建vue实例

## Vue常见指令（标签属性）

**基础指令**

v-text ：以纯文本方式显示数据，如遇到特特殊字符，如标签的"<"和">"会直接转移成Unicode码

v-html ：以html方式展示数据，如果有标签自动交给浏览器解析

v-show ： 控制元素的显示与否，核心以css的display属性用为控制方式，来显示或隐藏元素

v-if ： 控制元素的显示与否，核心以dom操作，动态创建或删除dom节点来实现。

v-model： 主要用于input表单项，实现双向绑定

数据->视图 ： 数据挟持（Object.defineProperty函数，对数据的set和get进行增强。

主要对set设置操作，增加界面dom的更新操作）【正因如此，Vue不支持IE8以下浏览器】

视图->数据 ： dom监听，捕获input事件（或change事件），找到绑定的对象属性，进行更新

**进阶指令**

v-bind ：用于对标签属性动态传参，参数值取自vue实例的数据。 可以用“:属性名”简写。

v-on ：用于对页面元素动态监听，参数值取自vue实例的方法。 可以用“@事件名”简写。

## Vue组件

Vue组件一共分为2种，“全局”和“局部”组件

全局组件

语法 ：Vue.component(组件名, 组件配置)

局部组件

语法 ：

new Vue({

...

components:{

// 组件名 ： 组件配置

}

})

注意：命名不要有大小写，多单词用“-”链接。

# Vue-cli（掌握）

## 全局安装Vue官方脚手架

>> npm install -g @vue/cli

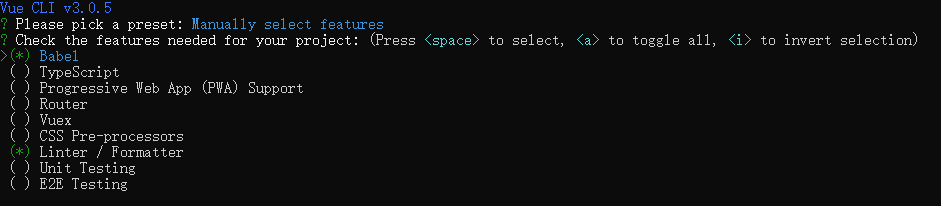
>> yarn global add @vue/cli （全局安装Vue cli3.0）

## 使用vue-cli创建项目步骤

### 命令行创建

// 创建基于webpack模板的项目，项目名字为名字为vue-app(\*名称不支持大写)

>> vue create vue-app



自定义功能配置包括以下功能：

* Babel**（默认）**

是否开启babel转义

* TypeScript

是否开启在Vue中支持TypeScript

* Progressive Web App (PWA) Support

是否开启PWA支持

* Router

是否自动集成路由

* Vuex

是否自动集成Vuex

* CSS Pre-processors

是否开启CSS 预处理（兼容）

* Linter / Formatter**（默认）**

是否集成语法规则和代码风格的检查工具，如ESLint。

ESLint with error prevention only——只检测错误。

ESLint + Airbnb config——独角兽公司的Airbnb，有人评价说“这是一份最合理的JavaScript编码规范”，它几乎涵盖了JavaScript的各个方面。

ESLint + Standard config——standardJs一份强大的JavaScript编码规范，自带linter和自动代码纠正。没有配置。自动格式化代码。可以在编码早期发现规范问题和低级错误。

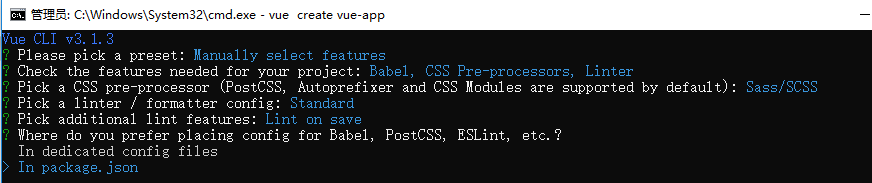
ESLint + Prettier—— Prettier 作为代码格式化工具，能够统一整个团队的代码风格。

* Unit Testing

是否集成单元测试解决方案，如JTest或者Mocha

* E2E Testing

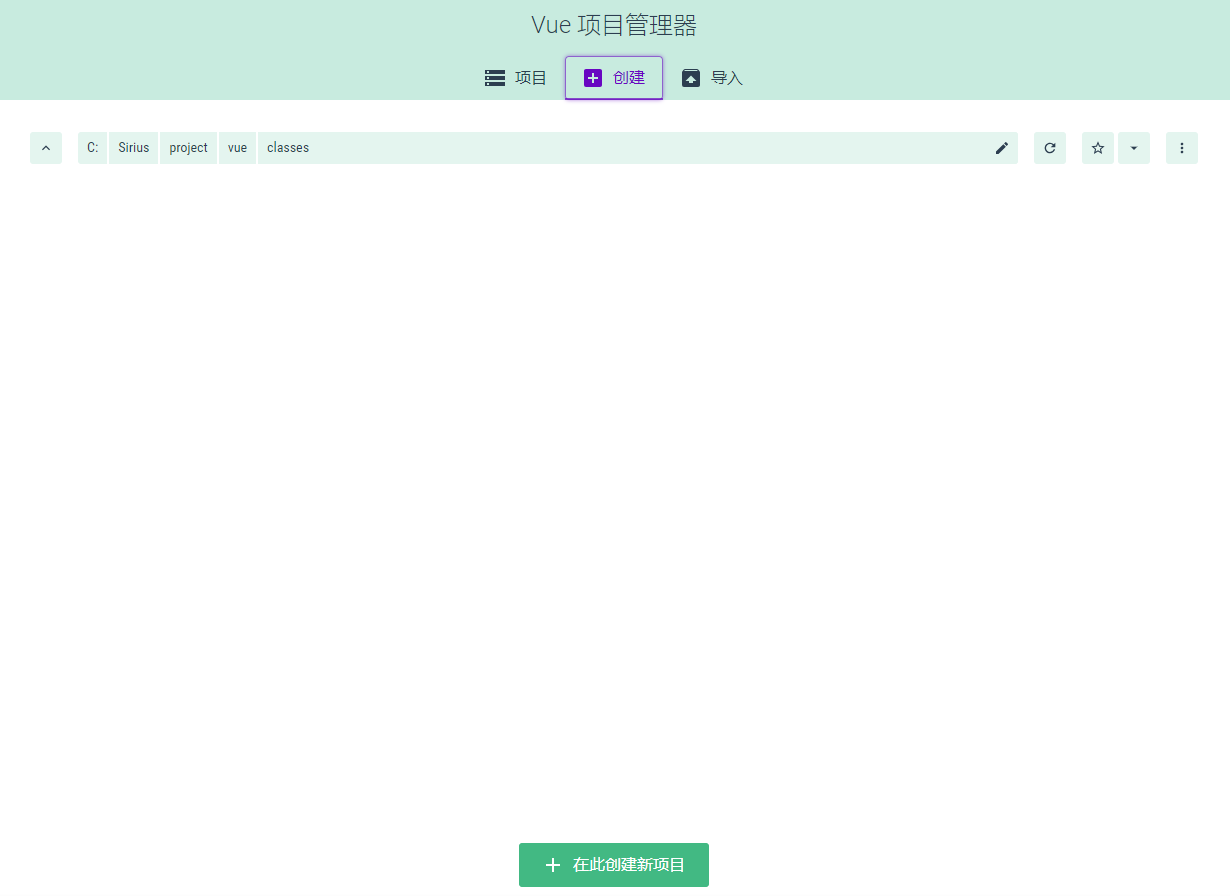
是否集成E2E（End-to-End）测试解决方案，如Cypress或者Nightwatch



配置都合成到package.json文件中。

### 可视化创建

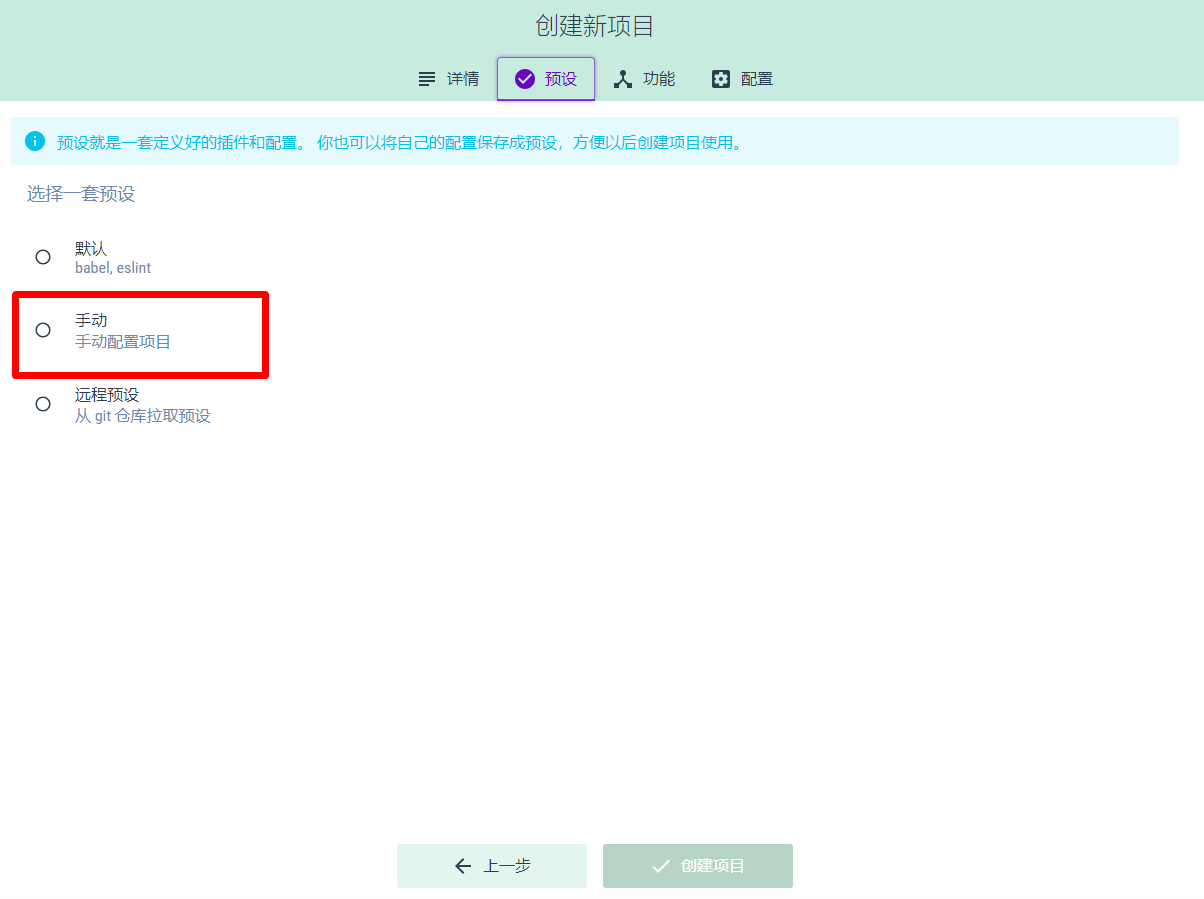
>> vue ui



进入管理界面，点击“在此创建项目”



填写项目基本信息。



选择手动配置。



基本功能，只需要选择babel和Linter/Formatter。

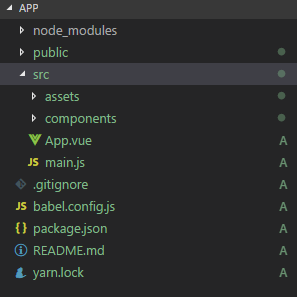


选择ESLint配置规范，设置检查时机未即时检查。





## 项目目录详解



## 单文件组件实践（写一个简单的单文件组件）

小结：

1. 使用Vue-cli官方脚手架搭建环境（熟悉脚手架选项配置）
2. 熟悉单文件组价编写

# VSCode代码片段设置

可以直接安装Vue VSCode Snippets。

1. 新建\*.vue文件，在内容中输入vbase回车，测试快速模板结果。

# Vue生命周期

生命总共分为4大阶段创建阶段、挂载阶段、更新阶段、销毁阶段，

共8个方法。

**创建前/后** : 在beforeCreate阶段，vue实例的挂载元素el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据data有了，el还没有。

\*创建阶段：主要是处理数据（主要：对数据进行挟持，把data对象上的数据挂到了this实例上）！！

**挂载前/后**：在beforeMount阶段，vue实例的$el和data都初始化了,单还没有挂载之前都是虚拟的demo阶段,data.message还未替换.在mounted阶段,vue实例挂载完后,data.message成功渲染.

\*挂载阶段：主要是合成数据和视图模板，先在内存完成虚拟dom解析，最后一次dom操作直接更新到页面。

**更新前/后**：当data变化后,针对视图发生更新前后触发beforeUpdata和updata方法。

\*更新阶段：主要是完成对视图的动态修改。

**销毁前/后**：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经结束了事件监听以及和dom的绑定，但是dom结构依然存在。

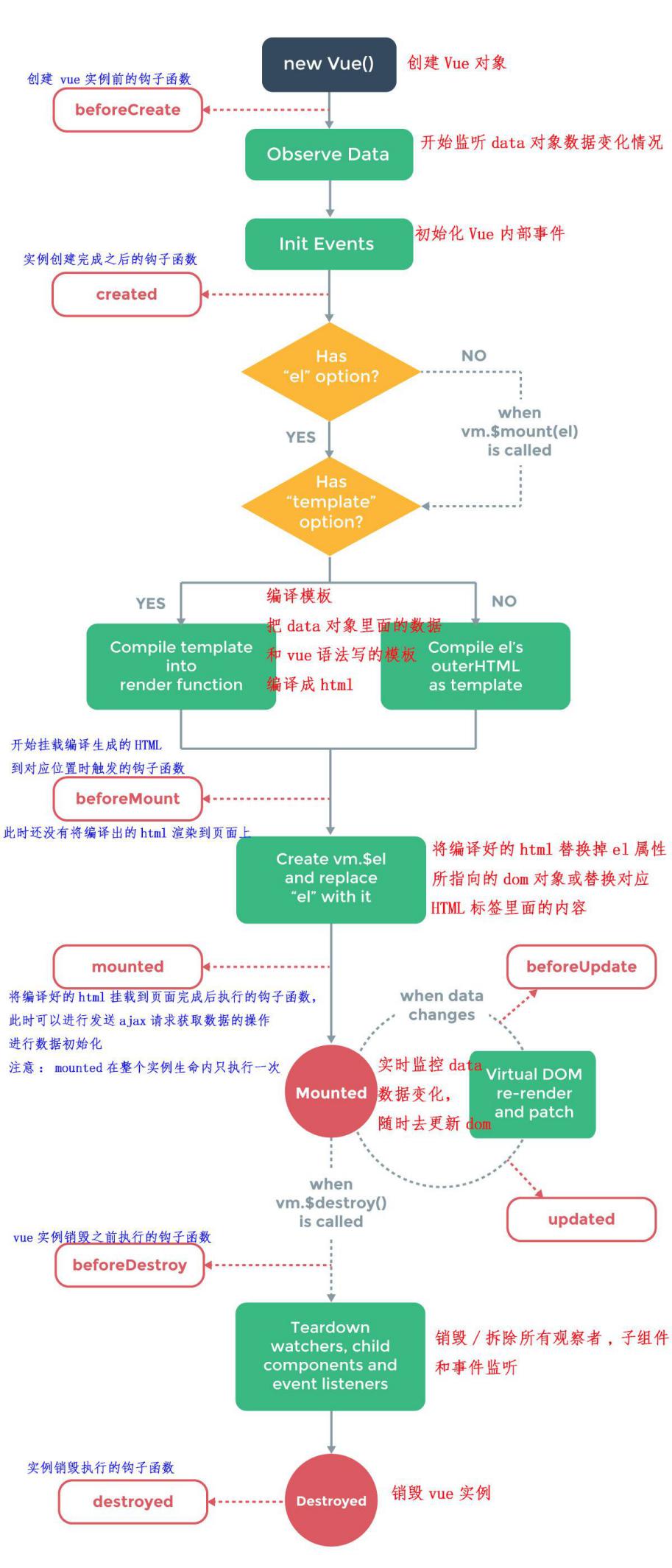
\*销毁阶段：主要是完成对组件的回收。

发请求最佳时间：大部分场景都会操作dom，就放到created生命周期钩子函数里面

created(){

this.$nextTick(() => { /\* code \*/ })

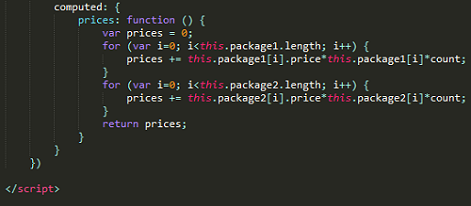
}



# 计算属性computed

在vue的计算属性里面可以完成各种复杂的逻辑，运算、函数调用等等这些都可以完成，只需要计算好了，最终返回一个结果就好了，同时，计算属性可以依赖多个vue实例的数据，只有任何一个数据发生变化，计算属性就会重新执行，视图也会更新。下面这个例子，展现了购物车内两个包裹物品的总价格：





当package1 或 package2中的商品有任何的变化，比如购买的数量变化，或者增加删除商品的时候，计算属性prices就会自动更新，视图中的总价也会自动变化。

  每一个计算属性都包含有一个getter和setter，我们上面的是使用计算属性的默认用法，知识用了getter来读取，在你需要的时候，也可以提供一个setter函数，当手动修改属性的值，就像修改一个普通数据那样是，就会触发setter函数，去执行一些自己定义的操作，例如下面的例子：

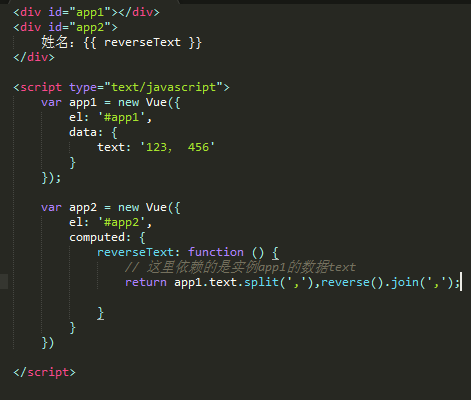


当执行app.fullName = ‘John Doe’时， setter就会被调用，数据fullName和lastName都会相对更新，视图也会更新。

  绝大多数情况下，我们只会用默认的getter方法来读取一个属性，在业务逻辑中，很少用到setter，所以在声明一个计算属性时，可以直接使用默认的写法，不必将getter和setter都声明。

  计算属性除了上述简单的文本插值意外，还经常用于动态的设置元素的样式名称class和内联样式，style，在使用组件时，计算属性也经常用来动态的传递props

  计算属性还有两个很实用的小技巧：第一个是计算属性可以依赖其他的计算属性，第二个是，计算属性不仅可以依赖当前vue实例的数据，还可以依赖其他实例的数据，如下：



这里创建了两个vue实例，app1和app2，在app2 的计算属性reverserText中，依赖的是app1的数据text，所以当text变化时，实例app2的计算属性也会变化，这种用法在多人协作开发的时候就可以用到，当你写的组件所用到的数据需要依赖他人的组件时，就需要使用这种方式。

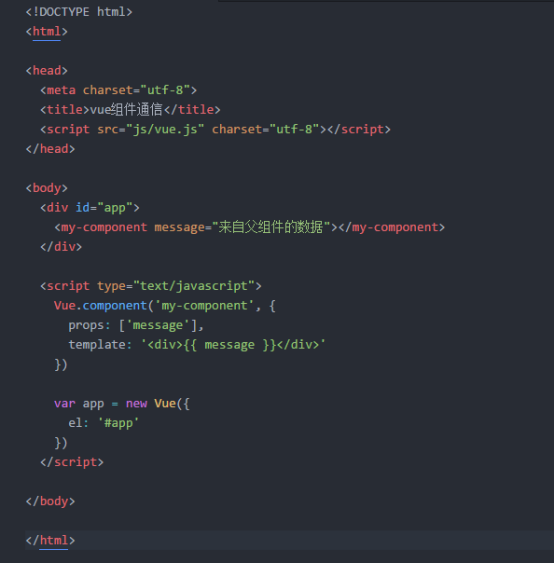
  当然，很多人也知道，我们调用methods里面的方法的时候，也可以取到和计算属性同样的作用，而且methods里面的方法还可以接收参数，使用起来更灵活，既然使用methods就可以实现，为什么还需要计算属性呢？原因就是计算属性基于它的依赖缓存的。一个计算属性，只有所依赖的数据发生变化时，它才会重新取值，所以text只要变，计算属性就不会更新，但是methods则不同，只要重新渲染，就会被调用，里面定义的函数也会执行。

  使用计算属性还是methods取决于你是否需要缓存，当遍历数组和做大量计算是，就需要用到计算属性。

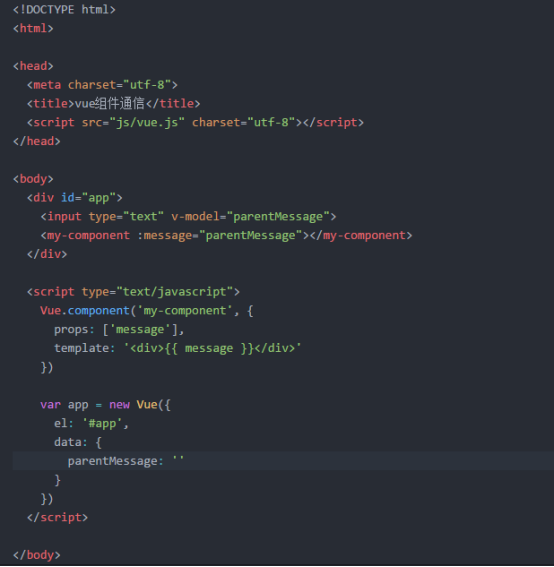
# 组件通信（掌握）

## 基本用法（父组件传给子组件）

Vue里面组件不仅是需要把模板的内容复用，更重要的是组件之间是需要通信的，我们先来说一种比较简单的情况，通常父组件包含子组件的时候，父组件要正向的向子组件传递数据和参数，子组件接收到后，可以根据参数的不同来渲染不同的内容或执行操作，这里是怎么来实现呢?  
 现在我们先来看看用props来传递数据的情况，在组件中，需要用选项props来声明需要从父级组件接收的数据，prop的值通常有两种，分别是字符串数组和对象，现在我们以数组为例子：我们构造一个数组，接收来自父级组件的message，并把它在组件模板中渲染。如下为示范代码：

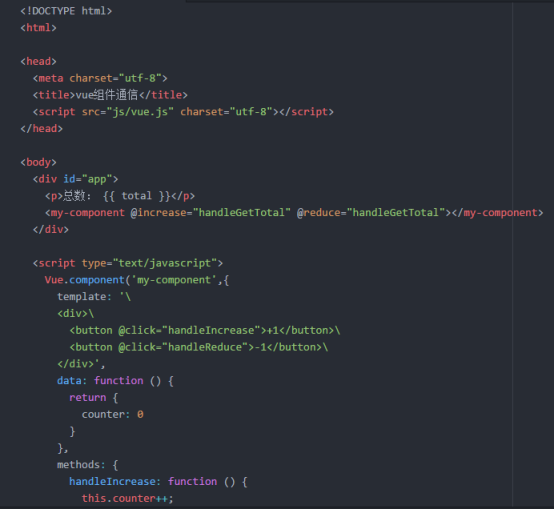


需要注意的是，props中声明的数据与组件data函数return的数据时有区别的，props的数据来自父级，而data中的是组件自己的数据，作用域是组件本身，这两种数据都可以在模板template、计算属性computed和方法mothods中使用。示例中的数据message就是通过props从父级传递过来的，在该组件的自定义标签上直接写props的名称，如果需要传递多个数据，在props数组中直接添加即可。  
注意：由于HTML特性不区分大小写，当使用DOM模板时，驼峰命名的props名称要转为短横线分隔命名  
很多时候，传递的数据不是死的，而是来自父级的动态数据，这个时候，需要使用指令v-bind来动态绑定props的值，当父组件的数据发生变化，也会传递给子组件，示例代码如下：



## 子组件传给父组件

当子组件需要向父组件传递数据时，就要用到自定义事件，v-on除了监听DOM事件外，还可以用于组件之间的自定义事件。在js的设计观察者模式里面有dispathEvent和addEventListener两个方法，Vue也要类似的一套模式，子组件用$emit()来触发事件，父组件用$on()来监听子组件的事件。当然，父组件可以直接在子组件的自定义标签上用v-on来监听子组件触发的自定义事件，示例代码如下：





示例中子组件有两个按钮，分别绑定了加1和减1的事件，并且在methods中实现了绑定了两个事件，在methods的事件实现中，通过$emit()方法，把它传递给父组件，父组件用@increase 和 @reduce接收，$emit()方法的第一个参数是自定义事件的名称，后面的参数是要传递的数据，父组件即可接收子组件传递的数据渲染页面，显示出加1过后的值。

# Vue-router

## 什么是前端路由

前端路由，即由前端维护的路由规则。常用的实现方式是利用url的hash，就是常说的锚点（#），每个url的hash映射一个对应的内容。

前端路由有很多的优点，例如页面的持久性，例如音乐网站，可以在播放歌曲的时候跳转到其他的页面，但是音乐没有中断。还有前后端彻底分离。

## vue-router基本用法

vue-router是vue的一个插件，路由不同的页面事实上就是动态加载不同的组件，具体使用方法如下：

新建一个vue项目目录，通过cnpm安装vue-router

cnpm i vue-router --save-dev

在入口文件main.js中引入路由插件，然后使用Vue.use()加载插件

import Vue from ‘vue’

import VueRouter from ‘vue-router’

Import App from ‘./app.vue’

**Vue.use(VueRouter)**

创建两个组件文件

//index.vue

<template>

<div>首页</div>

</tempalte>

<script>

Export default { }

</script>

//about.vue

<template>

<div>关于我们</div>

</tempalte>

<script>

Export default { }

</script>

在入口文件main.js里面，配置路由和组件的映射关系，每一个路由映射一个组件

import index from ‘./componet/index’

import aboutfrom ‘./componet/about

const Routers = [

{

path: ‘ /index ’,

component: index

},

{

path: ‘ /about ’,

component: about

}

]

然后，继续在main.js里面，完成配置和路由实例

Const RouterConfig = {

mode: ‘history’,

routers : Routers

}

const router = new VueRouter( RouterConfig )

new Vue({

el: ‘#app’,

router: router,

render: h => {

Return h(App)

}

})

在app.vue里面，添加一个路由视图<router-view><router-view>来挂载所有的路由组件：

<template>

<router-view></router-view>

</template>

<script>

expo t default {

}

</script>

<router-view>会根据当前路由动态渲染不同的页面组件

## 跳转

Vue-router里面有内置的组件<router-link>可以实现跳转，他会被渲染成一个a标签

// index.vue

<router-link to=’/about’>跳转到about</router-link>

## 路由嵌套

在路由中再定义子路由，叫做路由嵌套

1. var router = new VueRouter({
2. routes:[
3. {path:"/",component:index},
4. {path:"/product",component:product},
5. {path:"/about",component:about,
6. children: [
7. {path : "/" , component :us},
8. {path : "company" , component :company},
9. ]
10. }
11. ]});

# 课程总结

## 重点

* + - 1. 组件通信
      2. Vue-cli

## 难点

* + - 1. 组件通信

## 如何掌握？

* + - 1. 通过代码验证原理。
      2. 绘制思维导图表。

# 课后练习

**完成时间: 60分钟**

**简答题（手抄笔记）**

1. Vue组件之间的传值的方式。（难易指数：☆☆）
2. Vue-cli中怎么使用自定义组件。（难易指数：☆☆）
3. v-show和v-if指令的共同点和不同点。（难易指数：☆）
4. 什么是什么周期，vue生命周期总共有几个阶段，ajax在哪个阶段发请求？（难易指数：☆☆）

**操作题（电脑操作）**

1. 使用Vue-cli搭建项目
2. 使用Vue脚手架部署路由，手动实现多页面切换
3. 完成课题组件通信的代码
4. 使用Vue脚手架,完成一个登陆页面

# 面试题

1. 说出至少5中vue当中的指令和它的用法？
2. vue-loader是什么？使用它的用途有哪些？