**正课:**

**1. 自定义指令——了解**

**2. 过滤器**

**3. axios**

**4. 组件**

**5. 组件的生命周期**

**6. 组件化开发——重点&难点**

**1. 自定义指令——了解**

**Vue.js中除了13种预定义指令外，还需要增强HTML的其它功能，就需要添加自定义指令。**

**如何自定义:**

**1. 创建自定义指令:**

**Vue.directive("指令名",{ //创建一个自定义指令**

**inserted(elem){ //当当前元素被挂载到DOM树后自动执行**

**//elem 当前所在元素的DOM对象**

**//elem可调用一切DOM的API**

**}**

**})**

**强调: 定义指令时，指令名不要加v-前缀！**

**2. 使用自定义指令: <any v-指令名>**

**2. 过滤器(filter)**

**什么是: 接收原始数据后，执行再加工，才显示**

**何时: 原始数据不能直接使用时，都要用过滤器**

**比如: 性别(1,0->男，女)**

**钱(345.5-> ¥345.50)**

**时间(ms->字符串格式)**

**状态(10,20,30,40->未付款，已付款，已发货，已签收)**

**如何:**

**Vue官方没有提供任何预定义的过滤器，都要自己定义**

**1. 创建过滤器：**

**Vue.filter("过滤器名",function(value,形参1,形参2,...){**

**//value接收的是原始值**

**return 加工后的新值！**

**})**

**2. 使用过滤器:**

**只要绑定数据，都可用过滤器：**

**模型变量|过滤器名|......**

**模型变量|过滤器名(值1,值2)|......**

**强调:**

**1. 后一个过滤器获得的原始值，是前一个过滤器加工后的**

**2. v-text不能使用过滤器**

**3. axios:**

**什么是: 一个基于Promise的支持HTTP请求响应的函数库**

**为什么:**

**浏览器中向服务端发送ajax请求，4种:**

**1. 使用原生的xhr对象——繁琐**

**2. 使用jQuery中的$.ajax()——大材小用**

**3. 旧版Vue中，Vue官方自带了Vue-resource组件，专门发送xhr请求。——新版本中已经不推荐使用**

**4. axios，官方推荐的代替Vue-resource的，专门发送ajax请求的函数库**

**何时：只要在Vue中发送ajax请求，都用axios**

**如何: axios是一个单独的js文件，与平台无关。**

**即可用在浏览器中，也可运行在node.js中**

**在浏览器中：**

**1. 下载并引入axios.min.js文件，辅助下载并引入qs.min.js。**

**2. 调用axios函数，发起异步请求**

**get:**

**var res=await axios.get("url",{**

**params:{**

**请求参数1:值,**

**... : ...**

**}**

**}).then(function(res){**

**//res不是直接的响应结果**

**//res.data才是！**

**})**

**await和then选其一即可。**

**post:**

**先引入qs.min.js**

**var res=await axios.post("url",Qs.stringify({**

**请求参数1:值,**

**... : ...**

**}).then()**

**await和then选其一**

**4. 组件Component:**

**什么是: 拥有专属的HTML，CSS和数据的，可重用的页面独立区域**

**为什么: 重用！分工协作！松藕合！**

**何时: 只要发现独立的区域，都要封装为组件**

**今后的一个HTML页面，一定是由多个组件组成。**

**如何:**

**1. 创建组件:**

**Vue.component("组件名",{**

**template:"选择器", //代替了new Vue中的el:**

**data:function(){ //代替new Vue中的data:{ 模型变量 }**

**//每次都要调用函数，为本次组件创建专属的数据模型**

**return {//创建新对象的意思**

**模型变量: 值,**

**... : ...**

**}**

**},**

**其余都一样！**

**})**

**问题: 组件专属的HTML放在哪儿？**

**HTML5：<template>HTML片段</template>**

**template是专门保存一段隐藏HTML片段的元素**

**何时: 专门用于为组件保存HTML片段**

**规定: template中只能且必须有唯一的一个父元素包裹组件的片段。**

**如何:**

**1. 定义<template>元素包裹组件的HTML片段，还需要添加一个唯一的父元素**

**2. Vue.component("....",{template:"选择器"})，通过选择器找到对应的template元素**

**2. 在视图中使用组件:**

**一个组件其实就是一个可重用的自定义HTML标签  
 <组件名></组件名>**

**执行时，会用组件的template代替<组件名></组件名>**

**5. Vue实例/组件的生命周期：**

**鄙视: Vue实例/组件的声明周期共几个阶段，每个阶段做什么:**

**4个阶段:**

**1. 创建（create）阶段：创建组件对象，和模型数据对象**

**new Vue({ data: { ... } })**

**Vue.component("xxx",{ data:function(){ return { ...}}})**

**2. 挂载（mount）阶段：扫描真实DOM树，创建虚拟DOM树，并绑定模型中的数据到视图中**

**new Vue({ data: { } , el:"#app"})**

**Vue.component("xxx",{**

**data:function(){ return { ...}},**

**template:"#template"**

**})**

**3. 更新（update）阶段：只要模型数据发生变化，就更新组件的内容和DOM树**

**4. 销毁（destroy）阶段：主动调用了$destory()函数，销毁组件时**

**声明周期钩子函数：**

**什么是在组件的每个阶段都会自动执行的函数**

**何时: 如果希望在组件的不同阶段执行一项任务时**

**包括:**

**beforeCreate() $el:undefined data:undefined**

**created() $el: undefined data: { ... }**

**beforeMount() $el: 空DOM对象 data: { ... }**

**mounted() $el: 完整DOM树 data: { ... }**

**beforeUpdate()**

**updated()**

**beforeDestroy()**

**destroyed()**

**请求数据(axios.get())可放在created或mounted中均可**

**但是，如果有DOM操作，就必须放在mounted中**

**6. 组件化开发: ——重点&难点**

**将来拿到网页，先划分组件。**

**组件化开发，就是将大的网页划分为若干组件区域，每个组件区域都有专门的HTML，CSS和数据。**

**自定义组件，分为3种:**

**1. 根组件:**

**new Vue({**

**el:"父元素选择器",**

**data:{ 模型变量 }**

**})**

**2. 全局组件：可用在页面任何位置的组件**

**Vue.component("组件名", {**

**template:"template元素的选择器",**

**data:function(){**

**return { 模型变量 }**

**}**

**})**

**3. 局部组件: 子组件: 只能用于特定的父组件内**

**何时: 只要规定一个组件只能出现在指定的父组件内**

**如何:**

**1. 将全局组件降级:**

**var 子组件对象={**

**template:"template元素的选择器",**

**data:function(){**

**return { 模型变量 }**

**}**

**}**

**2. 在父组件中，添加components属性，包含子组件对象，并为子组件起名！**

**var 父组件={**

**... ...**

**components:{**

**"子组件名":子组件对象,**

**... : ...**

**}**

**}**